ANNUAL TECHNICAL REPORT TO THE DIRECTOR, ADVANCED RESEARCH PROJECTS AGENCY, FOR THE PERIOD 17 MAY 72-16 MAY 73

UNIVERSITY OF SOUTHERN CALIFORNIA

PREPARED FOR

ADVANCED RESEARCH PROJECTS AGENCY

MAY 1973

ARPA ORDER NO. 2223/1

ISI/SR—73—1

ANNUAL TECHNICAL REPORT

May 1972 – May 1973

prepared for the Advanced Research Projects Agency

INFORMATION SCIENCES INSTITUTE
UNIVERSITY OF SOUTHERN CALIFORNIA
4676 Admiralty Way/ Marina del Rey/ California 90291
(213) 822-1511

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author) USC Information Sciences Institute 4676 Admiralty Way Marina del Rey, California 90291	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3 REPORT TITLE

Annual Technical Report to the Director, Advanced Research Projects Agency, for the period 17 May 1972 to 16 May 1973

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)

Annual Technical 17 May 1972 to 16 May 1973

5 AUTHOR(S) (First name, middle initial, last name)

Keith W. Uncapher (Principal Investigator)

6 REPORT DATE	7a. TOTAL NO. OF PAGES ~~103~~ 101	7b. NO. OF REFS 69
8a. CONTRACT OR GRANT NO DAHC15 72 C 0308 b. PROJECT NO ARPA Order No. 2223/1 c. Program Code No. 3D30 and 3P10 d.	9a. ORIGINATOR'S REPORT NUMBER(S) ISI/SR-73-1 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	

10 DISTRIBUTION STATEMENT

Approved for release; distribution unlimited

11 SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209

13 ABSTRACT

This technical report summarizes the research activities performed by the Information Sciences Institute of the University of Southern California during the period 17 May 1972 to 16 May 1973 under Contract No.DAHC15 72 C 0308, ARPA Order No.2223/1, Program Code No.3D30 and 3P10 with the Advanced Research Projects Agency, Information Processing Techniques Office. The research is oriented toward the application of computer science and technology to problem areas of high impact.

The research program is composed of five elements: 1) Programming Research Instrument - the development and exploration of a major time-shared microprogramming facility; 2) Automatic Programming - the study of acquisition and utilization of problem knowledge for program generation; 3) Computer Software Assurance - methods to assess the viability of the security and protection mechanisms of operating system software; 4) Network-Supporting Research - packet-switched network communications technology development including voice, data, and image use in remote conferencing applications; 5) Programmed Automation - the application of computer science technology to automation of DOD high procurement areas, including impact studies and definition of development requirements.

Also reported is the development and operation of TENEX computer facilities supporting ARPA-sponsored Institute projects and some 400 external users via the ARPANET.

DD FORM 1 NOV 65 1473

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
2 ARPANET, control memory, microprocessor, microprogramming, microprogramming language, microvisor, MLP-900, operating systems, resource sharing, TENEX, time sharing, writable control memory						
3 automatic programming, BBN-Lisp and EL/1, domain-independent, human-engineered, interactive system, model acquisition, model verification, natural language, nonprocedural languages, nonprofessional computer users, optimization, problem solving, problem specification, process transformation, world knowledge						
4 access control, computer security, correctness, operating systems, permission functions, program proving, program verification, protection, software security, testing and evaluation, verification						
5 computer network, network accounting, network conferencing, network protocols, signal processing, speech processing, transparent networking						
6 automation, assembly, batch production, Computer-Aided Manufacturing (CAM), DOD procurement, Management Information System (MIS), manufacturing systems, Numerical Control (N/C), process control, production control, productivity, robotics						
7 paging, PDP-10, TENEX, time sharing						

a

RESEARCH PROGRAM

in the field of

COMPUTER TECHNOLOGY

ANNUAL TECHNICAL REPORT
May 1972 - May 1973

Effective date of Contract
17 May 1972

Contract expiration date
15 July 1975

Amount of Contract
$5,987,055

Principal Investigator
Keith W. Uncapher
(213) 822-1511

THIS RESEARCH IS SUPPORTED BY THE ADVANCED RESEARCH PROJECTS AGENCY UNDER CONTRACT NO. DAHC15 72 C 0308, ARPA ORDER NO. 2223/1, PROGRAM CODE NO. 3D30 AND 3P10.

INFORMATION SCIENCES INSTITUTE

UNIVERSITY OF SOUTHERN CALIFORNIA

4676 Admiralty Way/Marina del Rey/California 90291
(213) 822-1511

PERSONNEL

Research Staff:	**Robert H. Anderson**
	Robert M. Balzer
	Richard L. Bisbey II
	Thomas L. Boynton
	Jim Carlstedt
	Thomas O. Ellis
	Stuart C. Feigin
	Louis Gallenson
	Donald I. Good
	John F. Heafner
	Thomas N. Hibbard
	Ernest M. Hinds
	Robert E. Hoffman
	Gopal K. Kadekodi
	Nake M. Kamrany
	Bennet P. Lientz
	Ralph L. London
	Raymond L. Mason
	John T. Melvin
	Donald R. Oestreicher
	Robert H. Parker
	Elliot A. Ponchick
	Leroy C. Richardson
	Guner S. Robinson
	Keith W. Uncapher
	Albert L. Zobrist
Consultants:	**Michael Boretsky**
	Alexander F. Brewer
	Vernon Edwards
	Ronald Kaplan
	Martin J. Kay
	Gerald J. Popek
	Jack Rosenberg
Research Support Staff:	**Gerhard W. Albert**
	R. Jacque Bruninga
	George W. Dietrich
	Dorothy M. Fischer
	Oralio E. Garza
	Judy Gustafson
	Patricia A. Hagedorn
	Laurie J. Haron
	Chloe S. Holg
	Mary J. Jaskula
	Lloyd G. Jensen
	Rose L. Kattlove
	G. Nelson Lucas
	Jerry Pipes
	Judith E. Speer
	Sandra F. Whitaker
	Ruth White
Research Assistants:	**Robert W. Lingard**
	John M. Malcolm
	Nadine E. Malcolm
	David Wilczynski
	Martin D. Yonke
Student Aides:	**Raymond L. Bates**
	Danny L. Charlesworth
	Ronald L. Currier
	Jimmy T. Koda
	Kyle Lemmons
	Virginia E. Sato

Edited by Laurie J. Haron

Designed and illustrated by G. Nelson Lucas

CONTENTS

FIGURES

TABLES

ABSTRACT

This technical report summarizes the research activities performed by the Information Sciences Institute of the University of Southern California during the period 17 May 1972 to 16 May 1973 under Contract No. DAHC15 72 C 0308, ARPA Order No. 2223/1, Program Code No. 3D30 and 3P10 with the Advanced Research Projects Agency, Information Processing Techniques Office. The research is oriented toward the application of computer science and technology to problem areas of high impact.

The research program is composed of five elements: 1) *Programming Research Instrument* - the development and exploration of a major time-shared microprogramming facility; 2) *Automatic Programming* - the study of acquisition and utilization of problem knowledge for program generation; 3) *Computer Software Assurance* - methods to assess the viability of the security and protection mechanisms of operating system software; 4) *Network-Supporting Research* - packet-switched network communications technology development, including voice, data, and image use in remote conferencing applications; and 5) *Programmed Automation* - the application of computer science technology to automation of DOD high procurement areas, including impact studies and definition of development requirements.

Also reported is the development and operation of TENEX computer facilities supporting ARPA-sponsored Institute projects and some 400 external users via the ARPANET.

1 OVERVIEW

The Information Sciences Institute (ISI), a research unit of the University of Southern California's School of Engineering, was formed in May 1972 to do research in the fields of computer and communications sciences with an emphasis on systems and applications. The Institute, located off-campus, has sufficient autonomy within the University structure to assure it the freedom required to identify and engage in significant research programs.

At the end of the first year of operation, ISI's full-time professional research staff numbers 25. Total project support at ISI, including full-time staff, participating faculty and graduate students, and support personnel, is currently at 60 people.

A close relationship is maintained with USC academic programs through active cooperation between ISI, the School of Engineering, the Department of Electrical Engineering, and the Computer Science Program. Ph.D. thesis supervision is an integral part of ISI programs. Also, participating faculty and graduate students from other departments provide the interdisciplinary capabilities for Institute projects.

ISI has five major projects supporting ARPA/IPT research areas. These projects include the following elements:

PROGRAMMING RESEARCH INSTRUMENT: Development and exploration of time-shared microprogramming facilities to service, via the ARPANET, language processing, emulation, and special functional needs.

AUTOMATIC PROGRAMMING: Definition and contributing research in automatic programming, including problem acquisition, modeling, and code production.

COMPUTER SOFTWARE ASSURANCE: Studies and development of methodologies for system software security and protection, including software design concepts, formal program verification, and empirical measurement efforts.

NETWORK-SUPPORTING RESEARCH: Research and development of packet-switched network use in secure natural communications, including voice, image, and textual data; remote subroutine use, and special studies in network applications for DOD environments.

PROGRAMMED AUTOMATION: Definitional studies leading to programmable automation of job shop environments, including impact studies on DOD's procurements and recommendations for development programs required to realize such automated systems.

ISI has conducted several smaller projects related to more immediate DOD needs. These have included a study and comprehensive report proposing a means for total automation of military message services on the island of Oahu, Hawaii; work in the definition and implementation of ARPANET accounting practices; and definition and development of a general-purpose, high quality terminal display and hard copy facility.

ISI has implemented and maintained a major TENEX computing facility in support of Institute projects and other ARPA programs

via the ARPANET. This system is now the busiest node on the ARPA Network; operating at full capacity it has attracted some 400 users in its first 8 months of service.

The blend of talents, along with a full-time commitment to research by the staff, has resulted in the creation of a new institute well suited to contemporary research needs and required accomplishments.

The following is a report on research projects initiated in the first year of operation of ISI.

2 PROGRAMMING RESEARCH INSTRUMENT

Project Leader: **Thomas O. Ellis**

Research Staff: **Stuart C. Feigin**
Louis Gallenson
Robert E. Hoffman
Raymond L. Mason
John T. Melvin
Donald R. Oestreicher

Research Support Staff: **R. Jacque Bruninga**
George W. Dietrich
Oralio E. Garza
Mary J. Jaskula
Lloyd G. Jensen

Research Assistants: **John M. Malcolm**
Martin D. Yonke

INTRODUCTION

The Programming Research Instrument (PRIM) project is creating a fully protected experimental computing environment with continuous multi-user access. The I/O and user interaction facilities will be provided by Bolt Beranek and Newman's Inc. (BBN) TENEX time sharing system. The computational facilities will be provided by the MLP-900, a flexible and powerful microprogrammed processor developed by the STANDARD Computer Corporation. PRIM will provide a multi-access system in which each researcher can create his own specialized computing engine capable of being changed and adapted to his specific needs. This facility will be available to users through the ISI TENEX host configuration on the ARPANET.

The PRIM project will create an ARPANET-based sharable design environment, which will be used as a means of exploring computer architecture, language development, and special purpose processor design. These factors are all of particular relevance to DOD selection and use of computer equipment.

As with the first generation CPUs 10 years ago, most microprocessors now operate in a single-user environment. They have only minimal operating systems and are dedicated to a single application. This type of configuration makes the general availability of a microprogrammed resource both difficult and expensive.

The reasons for this situation are familiar from 10 years ago. First, most microprocessors have not been structured to allow for time sharing at the microprogramming level. Most are devoid of the proper protection and interrupt facilities to support this kind of operating system. Second, control memory is too expensive and, as a result, too small for users to be willing to give up any significant amount of core space to a time sharing system. Finally, the unfamiliar architecture of microprocessors is difficult to program and high-level language tools are rarely available.

These problems have been occupying the PRIM project for the past year. Solutions are

now at hand, and a time-shared microprocessor should be available to ARPANET users sometime in the late summer of 1973.

PRIM Hardware Resources

This system is based on two processors: the Digital Equipment Corporation (DEC) PDP-10, and the STANDARD Computer Corporation MLP-900 prototype processor. (See Figure 2.1.)

PDP-10.

The PDP-10 has a BBN paging box and runs the BBN TENEX time sharing system. The processor has 256K words of 36 bit memory, and is connected to the ARPANET. In order to simplify, and thus minimize, the microprogram supervisor (microvisor) in the microprocessor, all I/O will be done by TENEX. This includes files, terminal and network handling, swapping, and all other accesses to peripheral devices.

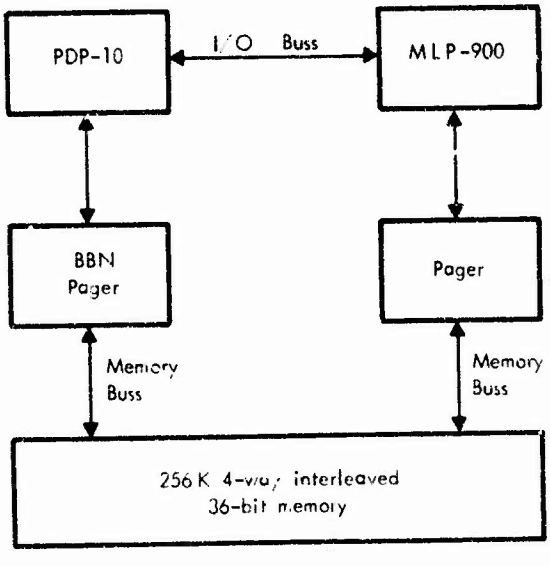

Figure 2.1 Basic PRIM configuration.

MLP-900.

The MLP-900 [1-3] is a vertical word microprocessor which runs synchronously with a 5MHz. clock. It is characterized by two parallel computing engines: the Operating Engine (OE) and the Control Engine (CE). The OE performs arithmetic operations and the CE control operations. The OE contains 32 36-bit general-purpose registers for operands, and 32 36-bit mask registers to specify operand fields. The CE contains 256 state flip/flops. The CE also contains a 16 word hardware subroutine return stack and 16 8-bit pointer registers. The writable control memory contains 4K words. There is a 1K 36-bit high speed auxiliary memory associated with the OE.

The OE and CE will execute in parallel if, and only if, a CE instruction follows an OE instruction in control memory. Programmer consideration of this feature usually is not required. However, if a program is executed entirely as OE-CE instruction pairs, the effective machine speed is doubled.

The speed and power of the MLP-900 may be conveniently characterized by its ability to emulate better known machines. Emulation of the IBM 360 machine language instructions would result in an estimated execution rate between 0.5 and 1.0 that of an IBM 360/65. A PDP-10 can be emulated at a rate almost double a KA10 CPU. However, in two high-level languages investigated, an estimated order-of-magnitude increase in execution rate of source statements can be attained by implementing those languages directly rather than emulating an intermediate target machine.

The MLP-900 is particularly well suited for investigating direct language emulation. It has the flexibility of a large (4,096 word x 36 bit) writable control memory. The basic architecture of the MLP-900 permits convenient expansion through the use of special purpose hardware language boards. These allow easy expansion and increased speed for specialized language processing tasks.

The environment around the MLP-900 further promotes easy experimentation and user access. The TENEX host system will not only provide complete I/O handling for the MLP-900, but also a developed, and in many cases familiar, environment for users. Together these two advanced systems should provide a most powerful and useful tool.

Hardware Effort

The hardware group has made modifications to the MLP-900 in two areas. First, the MLP-900 was interfaced to the PDP-10. The resulting modifications included memory and I/O buss interfaces. A paging box was also necessary to maintain compatibility with the BBN TENEX. Second, the proper protection features were added to support a microvisor and preemptive user scheduling. These modifications included a privileged microvisor mode and processor state protections.

Software Effort

The software group has worked on support software and microvisor design. Included in the support effort was the creation of a high-level General Purpose Microprogramming Language (GPM) for programming the MLP-900. In addition, the STANDARD Computer Corporation MLP-900 diagnostics have been translated into GPM. All new diagnostics and software will be done in GPM.

Microvisor State.

The introduction of a microvisor state has been of major importance to the PRIM project and, significantly, has affected other research projects in both the private and public sectors. Prior to this project, little had been done towards making the multitude of microprocessors potentially sharable resources. This initial experiment goes a long way toward making the microprocessor widely and inexpensively available.

Relevance and Impact

When the PRIM facility is complete, it will be possible to quickly and easily simulate new hardware architectures and designs in microprogrammed software. That is, software can be created for hardware not yet available, and hardware designs may be extensively used and changed before the cost of even a breadboard prototype is expended. This should both cut lead time and improve decisions connected with the special purpose hardware procurement cycle.

The unique PRIM facility will be a prototype for the future. As microprogrammed computers become more readily available, PRIM systems will become more important as a way to make the use of large microprocessor systems cost effective. The PRIM-developed techniques will be as important as the time sharing techniques developed 10 years ago.

HARDWARE

The hardware effort has been conducted in three broad areas:

1) reconfiguring the mainframe for necessary expansion, improved reliability, and better cooling;

2) interfacing the MLP-900 to the PDP-10 (including I/O and memory buss interfaces, and a paging facility);

3) creating a microvisor state within the MLP-900 with facilities for protection of the privileged resources and appropriate communication to change state.

The hardware design effort has been completed and is in the final stages of implementation. Modification to the mainframe is currently finishing and testing of the modified processor will start immediately. The extended software diagnostics will assist in this effort and insure the integrity of the processor. The final test phase will be the integration of the MLP-900 with TENEX in a dual processor configuration.

Mainframe Reconfiguration

The first hardware implementation task completed was the reconfiguration of the mainframe of the MLP-900 and the replacement of unreliable subsystems. The orientation of the original mainframe (see Figure 2.2) was deficient in cooling capabilities and did not have the expansion area necessary to house needed additional functions. The frame was rotated 90 degrees (see Figure 2.3) and extended to provide the additional back panel space required. The new configuration also rotated the logic boards from the horizontal to the vertical plane and facilitated an improved cooling system.

Two subsystems that proved unreliable were the power supplies and the control memory. The original control memory was fabricated with ECL memory chips (SC7301) produced especially for the STANDARD Computer Corporation. These memory chips consumed too much power and the fabricated

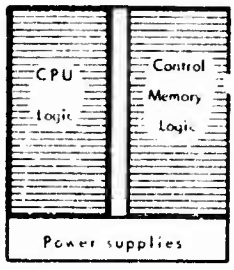

Figure 2.2 Original MLP-900 mainframe configuration.

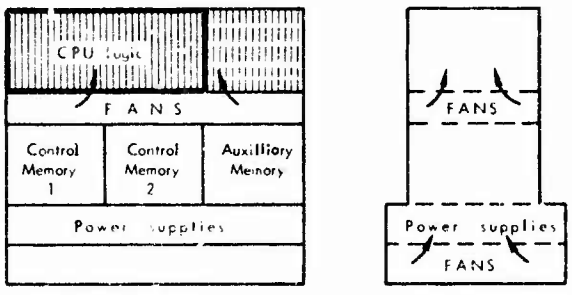

Figure 2.3 Reconfigured MLP-900 mainframe.

printed circuit memory boards proved to be of unreliable construction. This subassembly was replaced by an off-the-shelf memory system from Cogar Corporation which reduces power consumption at the expense of speed. The Cogar memory, which uses bipolar technology, has an access time of 60ns as compared with the 40ns specified for the STANDARD memory.

The decision to replace the control memory rather than repair the original one was based primarily on the guaranteed improved reliability. Other important considerations were: larger memory in a smaller area (the new memory is 4,096 words of 40 bits, compared to the original memory of 512 words of 32 bits); minimized heat dissipation problems; and off-the-shelf components for future expansion and/or replacement spares.

In addition to refurbishing the cooling system, the design also included a new power buss distribution, a power control panel, a facility for simple interface cabling, and a configuration which could be extended without revamping the existing mainframe. All these changes were implemented while keeping the original hack panel wiring and operator's panel intact.

Interface to PDP-10

The interface of the MLP-900 to the PDP-10 was accomplished by implementing a KA10 (CPU of the PDP-10) I/O buss connection to a register connected to the Exchange Buss of the MLP-900, and a memory buss from the ME10s (main memory of the PDP-10) to the Exchange Buss of the MLP-900 (see Figure 2.1). The Exchange Buss is the MLP-900 internal register buss that provides data paths for all registers and memories of the MLP-900.

I/O Buss Interface.

The I/O interface between the PDP-10 and the MLP-900 (see Figure 2.4) is designed and constructed to physically fit into the MLP-900 to achieve the following goals:

1) It is logically, electronically, and physically compatible with the PDP-10 I/O buss structure.

2) It is similarly compatible with a data line scanner via simulator to communicate the PDP-10 I/O buss information over a longer distance during initial checkout. An I/O buss simulator has been designed and implemented to allow a normal two wire (TTY) interface to the KA10. This allows complete checkout of the buss interface within the MLP-900 prior to connecting to the KA10 I/O buss directly.

3) It provides normal mutual communication for service requests and support functions between the PDP-10 and the MLP-900.

4) It provides positive preemptive control over the MLP-900 by the PDP-10 for emergency conditions.

5) It provides an Initial Program Loading (IPL) path for bootstrapping and emergency control over the control memory contents.

6) It provides the means for the PDP-10 to freeze, save, and restore the context of the I/O interface.

7) It provides a few direct indications of the MLP-900 status.

8) It provides adequate access paths to the registers for the MLP-900 to diagnose the I/O interface for error conditions.

9) It is designed for minimum interaction and interference between the MLP-900 and the PDP-10 in normal operations.

The I/O buss interface contains four registers. They are:

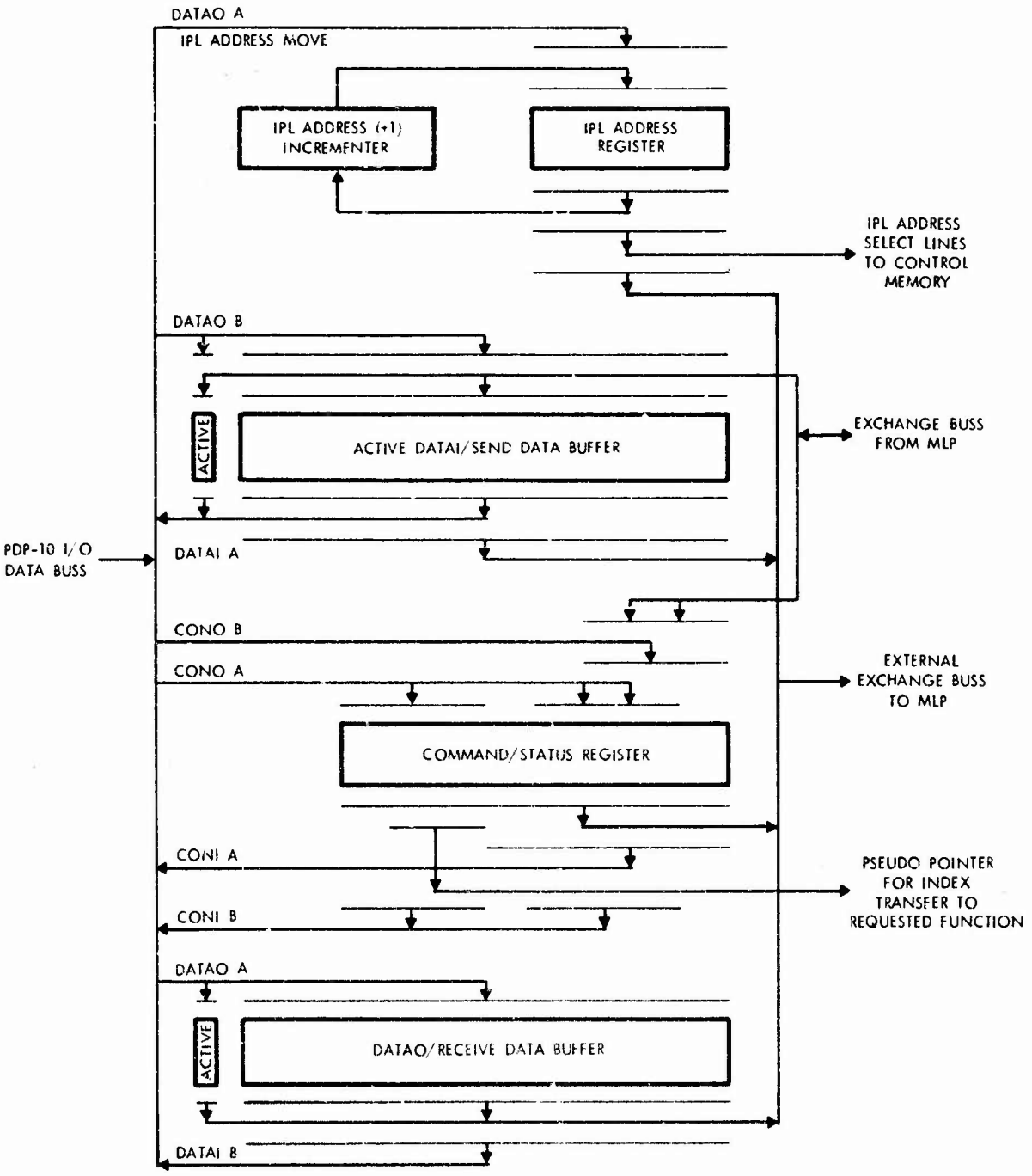

Figure 2.4 I/O Interface registers.

1) a data buffer, called DATAO/RECEIVE, to receive from the PDP-10 and send to the MLP-900;

2) a data buffer, called DATAI/SEND, to receive from the MLP-900 and send to the PDP-10;

3) a command and status register to accept commands from both, and to send out status and command information to both;

4) a counting register set by the PDP-10 and used to address control memory during Initial Program Loading.

Main Memory Buss Interface.

The system architecture of the MLP-900 integration into TENEX requires the sharing of a common main memory which results in a dual processor configuration. The functional requirements of the interface are:

1) to provide logic level compatibility between the MLP-900 and the ME10 core memory buss;

2) to provide a paging mechanism compatible with TENEX;

3) to provide a memory overlap access capability (streaming mode) to effectively increase main memory references when applicable.

The logic level compatibility problem is defined as a MECL II to the negative-voltage discrete-type technology used in the ME10 memory boxes. The solution required the development of a special circuit (translator) to drive the ME10 memory buss. To achieve minimum delays in accessing main memory, the buss is divided into four lengths (see Figure 2.5) and the translator is optimized to produce a 14ns circuit delay (see Figure 2.6).

The buss configuration yields a maximum cable delay of 35ns for a total maximum delay of 47ns, the latter within the specifications required by the streaming data transfers.

The MLP-900 pager (see Figure 2.7) is functionally identical to the BBN pager (TENEX). The paging task is one of mapping the user's virtual address space into real core memory address within protected pages of 512 words.

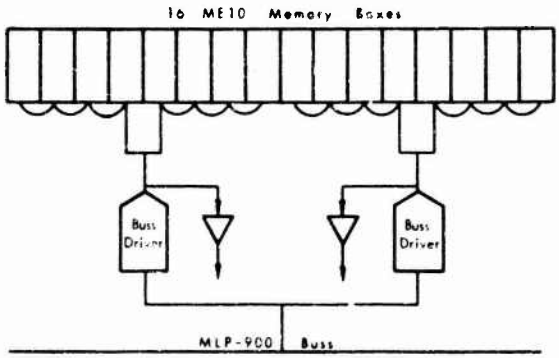

Figure 2.5 MPL-900 memory buss block diagram.

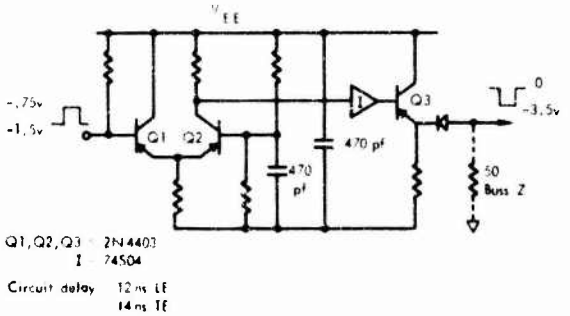

Figure 2.6 Memory buss driver schematic.

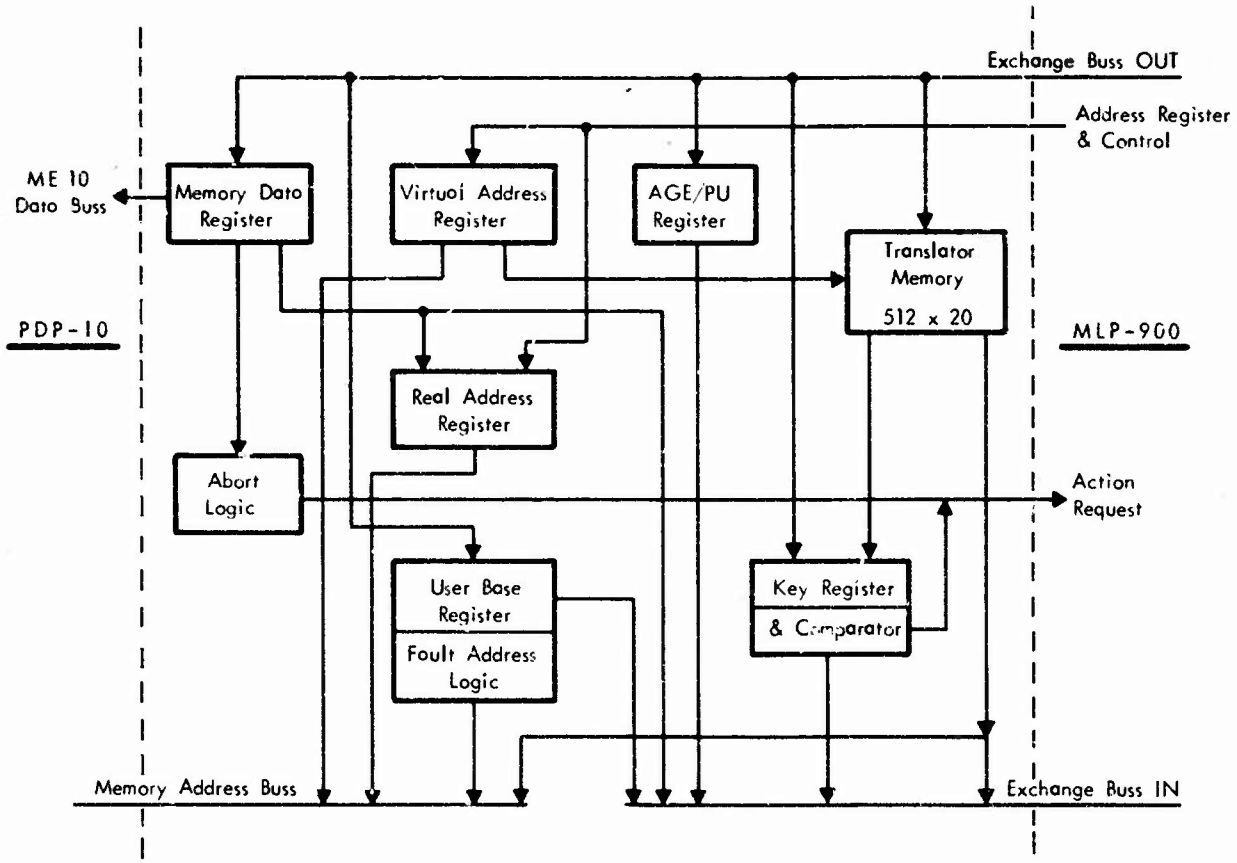

Figure 2.7 Simplified diagram of MLP Pager and main memory interface.

The pager is connected between the MLP-900 processor and the ME10 memory port. The address translation parameters are under control of the TENEX system and the parameters are fetched on demand by the pager logic from the predefined tables in main memory [4]. All fault functions performed in the BBN pager are performed by the new pager, including the Trap Status Word (TSW) which is handled by TENEX when nonrecoverable faults occur. There are three significant differences between the implementation of the MLP-900 pager and the BBN pager:

1) the microcode of the MLP-900 provides the major control functions when in paging fault (see Figure 2.8);

2) a transparent or real address mode is directly available to the MLP-900;

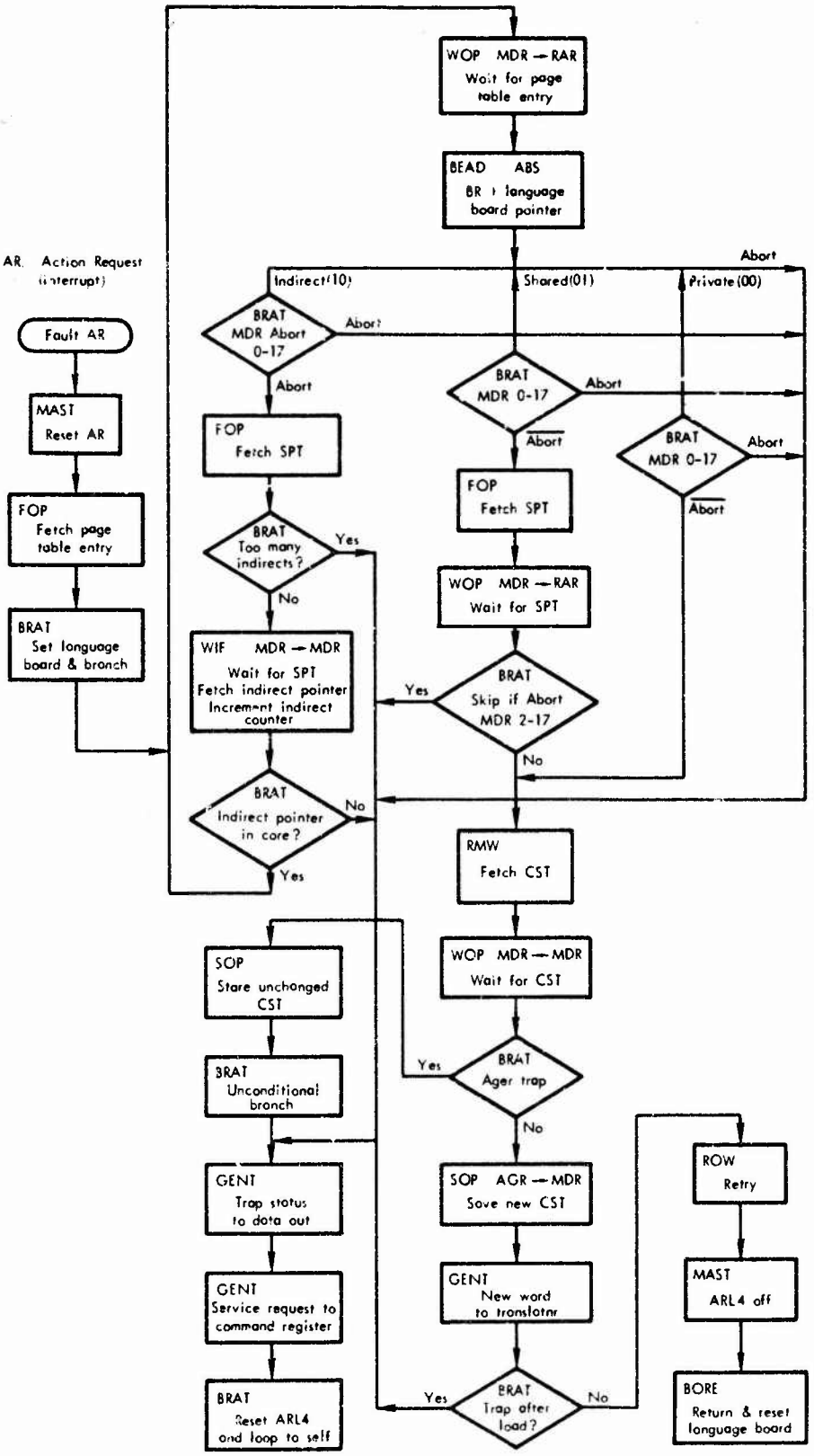

Figure 2.8 Paging fault algorithm.

3) the mapping process utilizes an addressable random access memory, rather than associative memory lookup.

The first two differences above are self-explanatory; the implementation of the mapping process is not. The 20 bit word of the translator memory is shown in Figure 2.9. The most significant 9 bits of the Virtual Address Register (VAR) are a pointer into the page memory whose contents contain an 11 bit Real Core Address (RCA). The 11 bit RCA is concatenated with the least significant 9 bits of the VAR to form a 20 bit address for main memory. (Only 9 bits of RCA are currently implemented for an 18 bit address.) The key field is compared with the contents of a key register (7 bits) to determine the validity of the parameters. The key register is incremented for each new user. Key register overflow causes a complete "erasing" of the pager memory. The microvisor is treated as another user.

As part of the main memory access, the MLP-900 interface was designed with a streaming mode. This mode will allow overlap memory operation to at least three levels, which produces an effective memory speed of 3MHz. The success of this capability may require additional design effort within the

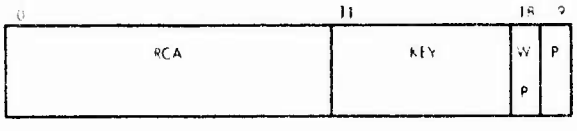

Figure 2.9 Translator memory word.

ME10 core boxes. This capability is being implemented by DEC for compatibility of the ME10 with the KA10 processor.

The streaming mode is designed to perform block transfer of data in and out of main memory with maximum memory overlap. The ME10s are capable of four-way interleave; successive memory requests can therefore be initiated without waiting for the completion of memory cycles. The MLP-900 operates at a 5MHz. instruction rate (design goal); with properly controlled delays (memory buss, memory speed, and MLP-900 interval buss) then, the memory requests can be generated as fast as ME10s can respond with address acknowledge (ADR ACK) and data is transferred in synchronism with ADR ACKs with the appropriate delays. The net effect is two separate pipe line operations: requests on the control lines of the memory buss, and data on the data lines of the memory buss. The MLP-900 is programmed to stop at the end of the block transfer (word equals zero), and discard the data remaining in the pipe.

Microvisor State

The microvisor state is synonymous with supervisor, kernel, or monitor mode of other CPUs utilized for time sharing applications. The microvisor has full access to all MLP-900 registers and instructions, whereas the user is limited in his access to protect the integrity of the microvisor. The new, wider control memory allowed 2 bits in each control memory location to be used to determine whether microvisor mode is to be enabled or not. The interpretation of the four possible codes is:

00 User Code: instructions executed are restricted to the user-machine subset.

01 Microvisor Entry Point: instructions executed have full microvisor privileges, and the user may transfer to these locations.

10 Microvisor Mode: same as 01, except the user may not transfer to these locations.

11 Data: nonexecutable.

Thirty-six bit data may now be stored in the new control memory. In the original STANDARD design, control memory was only 32 bits wide.

In addition to protecting control memory and privileged instructions, the microvisor state protects certain state flip/flops, the interrupt or action request hardware, and I/O and paging facilities.

Future Research Directions

As the testing proceeds to the point of user operation, the design effort will continue with additional functional user capabilities within the MLP-900. These functions are known as language board functions. The MLP-900 was originally designed to provide a language board to minimize the control code required to emulate a specific processor. Consideration is being given to generalize these functions to achieve CPU independence. This could provide the user with a more flexible MLP-900 configuration for structuring main memory, word size, interpreting instructions, stacking operands, and a variety of other functions still to be explored.

SOFTWARE

The software effort has been directed towards five specific goals:

1) design and development of a language facility for the MLP-900;

2) production of diagnostic programs for the MLP-900 processor;

3) extensions to TENEX to support the MLP-900;

4) development of a microvisor for the MLP-900;

5) design and development of user support functions for the MLP-900 facility.

The first four tasks have completed the design phase and are at various stages of implementation. The user support packages, which will include interactive program creation, and execution and debugging facilites, are in the early design phase.

MLP-900 Language Facility

Microprogrammed computers are typically characterized by small control memories for the storage of microprogrammed routines. These routines are used to implement the firmware instruction set for the target computer. The storage requirements and target computer instruction execution time considerations bring pressure on the microprogrammer to make optimal use of the microprocessor and associated control memory.

These conditions make microprogramming language designers and/or programmers tend towards a one-to-one correspondence between language statements and actual hardware microinstructions. As a result, microprogramming languages often look like classic assembly languages [5,6]. The General Purpose Microprogramming Language (GPM) provides the convenience and readability of a high-level language without preempting the flexibility and machine state control available in assembly code [7].

The primary goal of the GPM design was to produce a high-level language which did not preclude any coding options. In particular, as GPM was to be the primary language for the MLP-900, every control memory code had to be possible as language output. The language had to be amenable to diagnostic programmers, application programmers, and researchers. This was accomplished by combining the appropriate features of assembly code with the complimentary high-level features.

This goal was achieved by designing a language with careful consideration of the hardware instruction set. The language was constrained not to implicitly affect the machine state at runtime. These considerations provided freedom and low-level control for the programmer. The compiler needed some flexibility to allow for high-level language forms. This flexibility was provided by allowing the language to produce several microinstructions for each language statement.

GPM is complete and preliminary documentation is available. All MLP-900 coding is being done in GPM.

Some GPM statements look very much like assembly language. These statements correspond to the I/O instructions. The high-level statements fall into four categories of interest:

1) syntactic block structure;

2) hardware generalization;

3) multi-instruction statements;

4) expressions.

Syntactic Block Structure.

The low-level constraints in the GPM design precluded the implementation of any dynamic storage allocation, or even of an operand stack. As a result, the block structuring in GPM may be considered to be a compile-time artifact. None the less, by using the block structure syntax in a most rudimentary way, the resulting language has been rendered more tractable and comprehensible.

The user may rename any memory cell at the top of a block. Any synonym defined at the start of a block is undefined at the end of the block. This allows procedures to give mnemonic names to parameters and temporary registers. Additionally, the practice of declaring registers at the top of a block renders the block's scope instantly apparent. This block-structured synonym facility, if exploited properly, can produce more readable programs.

Control statements. The block structure syntax is used to define the scope of IF and DO statements. The DO statement heads a block which will be iterated upon indefinitely. The method of exiting a DO block is the BREAK statement.

The above examples of applications of block structure syntax demonstrate how the concept can be useful in a semantically simple language. One could even restate the GPM design goal: design a language which is syntactically rich and semantically poor.

Hardware Generalization.

One of the more classic functions of a programming language is to provide a complete set of functions, where the hardware may not. For instance, on a computer with only a jump on less than zero, the programming language would provide all eight possible jumps relative to zero. GPM attempts to do this in all

cases where it is possible, without violating the design constraints.

This type of language feature allows the programmer to ignore some of the intricacies of the hardware. However, if a GPM programmer wishes, he may stick to the GPM subset which corresponds to the hardware MLP-900 instructions.

Multi-instruction Statements.

Some GPM statements will always compile into several hardware MLP-900 instructions. These are common functions with which the user should not have to concern himself.

In order to transfer data from the operating engine to the control engine, the Exchange Buss (XBUS) is used. This requires an OE-CE instruction pair to be executed in parallel, each instruction moving information to or from the XBUS and internal registers. Therefore, all interengine assignments require the generation of two instructions.

The GPM statement

```
CE0 ← R0 ;
```

will compile into*

```
XBUS ← R0 ;
CE0 ← XBUS ;
```

These multi-instruction generating statements make programs shorter and thus easier to read. The information not explicitly stated is essentially redundant in the above example. Thus, brevity is achieved without the introduction of obscurity.

* As GPM is the primary MLP-900 language, it compiles into the subset of itself which corresponds to actual hardware instructions.

Expressions.

Expressions are so common in high-level languages, it might seem out of place to devote space to them here. However, the constraints on the GPM design complicate the compilation of expressions. GPM is not allowed to make any implicit changes to the machine state at runtime. This precludes the introduction of temporaries to evaluate expressions. Two brief examples will demonstrate how expressions are to be handled.

Arithmetic expressions. The GPM statement

```
R0 ← R0 AND R1 + R2 ;
```

will compile into

```
R0 ← R0 AND R1 ;
R0 ← R0 + R2 ;
```

However the GPM statement

```
R0 ← R0 AND (R1 + R2) ;
```

will not compile, for lack of a temporary for (R1 + R2).

Boolean expressions. The GPM statement

```
F0 ← (F1 AND F2) or (F3 AND F4) ;
```

will compile into

```
IF F1 AND F2 THEN GOTO +3 ;
IF F3 AND F4 THEN GOTO +2 ;
IF NOT (F0 ← FALSE) THEN GOTO +2 ;
F0 ← TRUE ;
```

Boolean expressions will always compile as the program counter can be used as a temporary Boolean value. The expression evaluation in GPM leaves much to be desired, but it was felt that when the expressions worked, they

were so superior to the assembly code alternative that they would be included in spite of themselves.

Diagnostic Programs

The main thrust of the diagnostic task has been to convert the STANDARD Computer Corporation assembly language diagnostics into GPM. Diagnostics have also been necessary for the new facilities added to the MLP-900 by the hardware group. These include both the microvisor protection facilities and the PDP-10 memory and I/O buss interfaces.

The production of the complete diagnostic system for the MLP-900 requires three tasks to be completed:

1) conversion of STANDARD assembly language diagnostics into GPM;

2) creation of new diagnostics for new hardware facilities; and

3) development of a diagnostic supervisor to load and run the diagnostics.

Conversion of Existing Diagnostics.

The conversion of the STANDARD diagnostics into GPM took several steps:

1) the diagnostics were assembled on the STANDARD assembler on the IC-4000;

2) the assembly output on 7-track magnetic tape was moved to the PDP-10;

3) the hexadecimal code was extracted from the output file, converted to octal, and the octal was converted to GPM;

4) the new GPM source statements were inserted into the existing source file to replace the old STANDARD assembly language statements;

5) the new diagnostics were compiled in GPM.

This process gave the PRIM project a readable set of basic processor diagnostics. The conversion process was easier to develop than rewriting the 8,000 lines of STANDARD diagnostics by hand.

New Diagnostics.

The new diagnostics are being produced in GPM. The highest priority new diagnostics will be the I/O buss ones, as no user may use the MLP-900 until the I/O buss interface is checked out.

Diagnostic Supervisor.

Two diagnostic supervisors are under development. The first, which has been in use for several months, is the I/O buss simulator system. The system uses the I/O buss simulator over a teletype line. This allows communication with the PDP-10 without looking directly to the PDP-10 I/O buss. This enabled checkout to begin on the MLP-900 before the final checkout of the I/O buss interface.

The new diagnostic supervisor will use the real I/O and the new TENEX system call discussed below.

TENEX System Changes

To maintain reliability and continuity in the existing TENEX service, changes to the TENEX operating system core are being kept to a minimum. The PRIM project has designed one TENEX system call to enable a privileged user program direct access to the MLP (see Figure 2.10).

Microvisor

A microvisor has been designed to communicate with privileged user programs. This

```
        Returns:

        +1    Error - error code in Ac 1

        +2    Successful

        INPUT              RESULT

Ac 1  Ac 2  AC 3

Mode  0     0     Assign MLP - Map MLP core starting at 500000
                  Error return if MLP not available
                  Return page count in Ac 1
                  Modes:
                  1     Input M on interrupt
                  2     Do nothing on interrupt
0     -1    -1    Status MLP - Returns:
                  Ac 1:      CONI DEVA,1 (last interrupt DEVA)
                  Ac 2:      DATAI DEVA,2 (last interrupt DEVA)
                  Ac 3:      CONI DEVB,3 (last interrupt DEVB)
                  Ac 4:      DATAI DEVB,4 (last interrupt DEVB)
0     Code  Data  Output to MLP
                  Code:
                  B35  ON
                  B34  0-CONO     1-DATAO
                  B33  0-DEVA     1-DEVB
                  B32  0-Return Immediately
                       1-Wait for MLP Interrupt
0     0     0     Status MLP - Returns:
                  Ac 1:      Interrupt Count (DEVA)
                  Ac 2:      Interrupt Count (DEVB)
-1    0     0     Release MLP - This is called by RESET
```

Figure 2.10 MLP JSYS (TENEX system call).

microvisor initially will have only two functions: to load context into the MLP-900 processor, and to dump processor context into the PDP-10 core memory.

Future Research Directions

Two areas remain for the completion of the MLP-900 program facility: a more sophisticated microvisor, and a user support facility.

Microvisor.

The target microvisor:

1) will handle page faults in order to allow for demand paging of the target address space;

2) will support I/O requests from MLP-900 user programs to TENEX; and

3) will contain the necessary hooks for the debugging system.

User Support Facility.

The user support facility will first provide a convenient debugging environment for MLP-900 programs. Once produced, it will be necessary to support MLP-900 programs in several configurations. A partial list of possible configurations include the MLP as a subroutine (function box) to a PDP-10, the MLP running in parallel to a PDP-10 user program, and the MLP using the PDP-10 as memory.

Upon completion of the PRIM facility, there are essentially two areas in which the PRIM project expects to continue. The first is in the development of a facility for high-level language interpretation. It has been shown by the B1700 group at Burroughs-Goleta and in several other studies that interpreting high-level languages can produce far more significant computational efficiencies than simple emulation of existing hardware. Along this line, two specific proposals are being considered: a Lisp execution engine based on Peter Deutsch's work at Xerox-Palo Alto Research Center in support of the Automatic Programming project; or a Pascal execution engine in support of the Computer Software Assurance project.

The second area, and of no less import, is the interaction with and support for ARPA Network users who may avail themselves of the PRIM facility. We hope that the unique facility being provided through the effort of the PRIM project will be useful not only to ISI, but also to other governmental contractors throughout the ARPANET.

MLP-900 User Programming Facility.

The design and development of a high-level programming language are considered the first of two minimum requirements for efficient user utilization of the MLP-900. The second requirement is for a symbolic debugging facility of sufficient power to allow a user of moderate experience the ability to meaningfully breakpoint and patch MLP programs.

The debugging mechanism is being studied in parallel with the current ongoing hardware development. It is expected to capitalize heavily on the experience gained during the integration of the MLP/PDP-10 system. The early phase will rely on a modified version of a currently available DDT that is a TENEX system programming tool.

Considerable attention will be paid to acquiring internal measurements of system performance as modifications are made to the scheduling and core managing algorithms. Both KA10 and MLP performance will be monitored. Modifications designed to improve performance will be investigated and verified by measurement under controlled and non-controlled loading. It is considered desirable to monitor and verify overall system evolution through direct measurement. A user-mode statistics gathering and analysis package will be developed to provide an adequate portrayal of system performance.

REFERENCES

1 STANDARD Computer Corporation, *MLP-900 Multi-Lingual Processor - Principles of Operation*, STANDARD Computer Corporation, Technical Publications, Santa Ana, California, 1970.

2 Lawson Jr., H. W., and B. K. Smith, "Functional Characteristics of a Multi-lingual

Processor", *IEEE Transactions on Computers*, Vol. C-20, No. 7, July 1971, pp. 732-742.

3 Guffin, R. M., "Microdiagnostics for the STANDARD Computer MLP-900 Processor", *IEEE Transactions on Computers*, Vol. C-20, No. 7, July 1971, pp. 803-808.

4 Strollo, T. R., J. D. Burchfiel, and R. S. Tomlinson, *Technical Details of the BBN Pager Model 701*, Bolt Beranek and Newman, Inc., Cambridge, Massachusetts, July 1970.

5 Dubbs, E. W., R. L. Parsons, and J. E. Peterson, "A Microprogram Design System Translator", in Sixth Annual IEEE Computer Society International Conference, *Digest of Papers*, San Francisco, California, September 12-14, 1972, pp. 95-98.

6 Berndt, H., "Microprogramming with Statements of Higher-Level Languages", in *Fifth Annual Workshop in Microprogramming*, Urbana, Illinois, September 25-26, 1972, pp. 76-80.

7 Eckhouse, R. H., "A High-Level Microprogramming Language", *AFIPS Conference Proceedings*, 1971 Spring Joint Computer Conference, Vol. 38, AFIPS Press, Montvale, New Jersey, 1971, pp. 169-177.

PUBLICATIONS

1 Oestreicher, Donald R., *A Microprogramming Language for the MLP-900*, USC/Information Sciences Institute, RR-73-8, June 1973.

ACTIVITIES

1 Oestreicher, Donald R., "A Microprogramming Language for the MLP-900", ACM Sigplan Sigmicro Interface Meeting, New York, May 30-June 1, 1973.

DOCTORAL THESES IN PROGRESS

1 Yonke, Martin D., "A Semantic Approach for On-line Program Development".

3 AUTOMATIC PROGRAMMING

Project Leader:	**Robert M. Balzer**
Research Staff:	**Robert E. Hoffman** **Albert L. Zobrist**
Consultants:	**Ronald Kaplan** **Martin J. Kay**
Research Support Staff:	**Chloe S. Holg**
Research Assistants:	**Robert W. Lingard** **Nadine E. Malcolm** **David Wilczynski**

INTRODUCTION

ISI's work on automatic programming has centered on three main activities. First, we worked with ARPA to help establish the direction and scope of the ARPA program in automatic programming. Towards this end, discussions were held with most of the prominent people in the community, both ARPA and non-ARPA contractors, and their views were solicited on the current state of the art and the direction future research should take. These views were integrated with our perception of the field to form a coherent picture and approach to automatic programming. These findings were presented in a report [1] which was sent to all those participating in the study. It was the subject of numerous discussions around the country, and acted as a straw man for an ARPA approach to automatic programming in two meetings of the community which were held to arrive at a consensus view.

Second, we have developed a language-independent interface between a user and his programming language which makes available to him the programming environment of Bolt Beranek and Newman's (BBN) LISP. This is a significant advancement which enables users of a wide variety of programming languages to use a powerful, common, well-engineered environment for programming. Furthermore, the interface has spurred the credibility of our revised proposal for ARPA's national program: namely, the creation of a facility for software production which will contain a common set of tools available to users of different languages running on different machines. This facility will utilize the ARPANET to establish communication between tools running on diverse machines, and will focus attention on the tools needed to increase productivity and the technology needed to effectively integrate them into a well-engineered system.

Third, formulative work has continued in the comprehension of human dialogue and the extraction of knowledge of new problem areas from their descriptions. We thus are investigating the possibility of domain-independent systems which can be taught (by an expert in that area) about new domains in terms of the objects which exist in that domain, the relationships between these objects, the transformations which can occur within the domain, and the constraints which must be satisfied. This knowledge will then be used to transform procedurally-specified tasks

within the domain into efficient computer programs which perform these tasks. The importance of this effort lies in the attempt to build a computer system which learns about new areas of competence in terms appropriate to that domain, and uses this knowledge to fill in the details of a high-level procedural specification for a task in that domain.

THE AUTOMATIC PROGRAMMING REPORT

As presented in the report [1], the goal of automatic programming is simply the effective utilization of computers, which implies both simplicity of use and efficient application of computing resources. Thus, automatic programming is the application of a computing system to the problem of effectively utilizing that or another computing system in the performance of a task specified by the user. Although this is certainly what is meant by automatic programming, this definition does little to restrict the set of applicable computer systems included in the automatic programming domain.

We therefore chose to restrict the definition by emphasizing the use of such systems by those who are not professional programmers. This emphasis necessitates the use of less precise and more problem-oriented descriptions of the computing task, and increases the amount of translation required from the system to turn such specifications into efficient programs.

Problem-oriented Model

Towards this end, we conjecture that the solution of every computable problem can be represented entirely in problem-domain terms as a sequence (which may involve loops and conditionals) of actions in that domain which affect a data base of relationships between the entities of the domain. Included, either as part of the data base or as a separate part of the model, is the history of the model (i.e., the sequence of actions applied to the model). This logically completes the model and enables questions or actions involving historical information to be handled. In a strong sense, such a solution is a direct simulation of the domain. The system models at each step what would occur in the domain.

The important part of the above conjecture is that any computable problem can be solved, thence described, in problem-domain terms. The solution, then can be divided into two parts: an external and an internal part. The external part is the problem specification given by the user in completely domain-specific terms. The requirement for such users is no longer a comprehensive knowledge of computers, but rather the ability to completely characterize the relevant relationships between entities of the problem domain and the actions in that domain. In addition, such users should have a rough awareness of the problem-solving capability of the system so that they can provide additional help where needed in the form of more appropriate macro-actions, recommendations about the use of certain actions, and/or imperative sequences which will solve part or all of the problem in problem-related terms.

The internal part is first concerned with finding a solution in problem-related terms, if this has not already been provided by the user. Second, this part is concerned with finding efficient solutions given the available

computing resources. Such optimizations occur at two levels beyond what is normally considered optimization. First, at the problem level, recognition that certain entities and/or relationships are irrelevant enables their removal from the model. Second, since only part of the state of the modeled domain is required, and only at certain points in the solution process, rather than simulating the model completely at each step, the system can employ alternative representations which require less maintenance and which either directly mirror the required part of the domain state or allow such parts to be computationally inferred. Such representations may also enable more direct solution of the problem. It is these optimizations which form the main distinction between the code-generation part of an automatic programming system and current state of the art compilers.

Thus, the definition of an automatic programming system is one which accepts a problem in terms of a complete model of the domain, which obtains a solution for the problem in terms of this model, and which produces an efficient computer implementation of this solution in the form of a program.

OTHER APPROACHES

Unfortunately, this view did not represent a consensus of the automatic programming community. Three divergent approaches to the problem (see below) emerged from the two meetings held to discuss these issues.

Automatic Programming Focus

The first meeting started with a broad discussion of automatic programming and what it covered. The general feeling appeared to be that automatic programming, as defined in the report [1], was too broad and covered too much ground. Because of this, discourse between the different disciplines involved would be difficult, if not impossible. It was, however, suggested that automatic programming was a good organizational device to focus attention on a common goal. That goal is increasing the accessibility of computers to users, especially to nonprofessional computer users.

Nonprofessional Users.

Although it was difficult to describe the nonprofessional users (e.g., whether or not they possessed procedural knowledge), it was generally agreed that their main attribute was that they would be working with the system through a model. Hence, the whole notion of modeling appears to be at the very core of automatic programming. The ease with which users can express their tasks in terms of the model, their ability to view the system's actions in terms of this model, and the ability to define, modify, and enhance models will be critical issues in automatic programming systems.

It was pointed out that the term *complete model*, as defined in the report, was unfortunate, for although the behavior of the user's task could be completely described in terms of this model, the model itself must be treated by the automatic programming system as incomplete and dynamically subject to change by the user during the course of his problem specification.

Further attempts to describe the nonprofessional users yielded the conclusion that there were at least two classes of such users. First, there are the experts in the field who would define basic models applicable to that field,

and build up within the system a large body of knowledge in suitable form. In a sense, they are more constructors of systems than end users. The second class, end users of such systems, would either parametrically use the supplied models or would modify and enhance these models for specialized usage. It was generally agreed that were a suitable automatic programming system available, experts in wide varieties of fields would be more than willing to devote their energy towards building the basic models for that field.

Discussion now returned to describing automatic programming itself, and it was brought out that the report was very heavily biased toward discussion of the system as if it were well-defined thing. Rather, it is a research domain, one that is difficult to define. At this point it was mentioned that specific systems of the type envisioned were being built or already existed. As such, what was the new research area? Surely it was not further systems of this type themselves. Rather, automatic programming is the methodology of building such systems. That is, the automatic development of special purpose systems.

Three Approaches

At this point, one of the key considerations emerged. Part of the trouble in discussing automatic programming systems was that different approaches to the problem were being discussed. There appeared to be three rather distinct approaches to the problem, which are called the low, medium, and high roads. These three approaches factor nicely as to the amount of involvement of models, the technology upon which they depend, and the time scale required for their development.

Low Road.

The low road is the most imminent and the one most closely associated with computer science. It is based on the paradigm that automating programming is best done by providing the programmer with tools that enhance his capabilities. These tools are of three basic types. The first is high-level languages which enable him to describe his problems in more appropriate terms and reduce the amount of translation he must perform. It includes language development at both the control and data level, and library systems which enable the programmer to assemble his system from prepackaged modules. A second major tool is optimization -- global optimizations which would allow the programmer to utilize these high-level languages and prepackaged modules without concern for efficiency, with the knowledge that they can later be automatically optimized. These optimizations include and critically depend upon the system's choice of data structures and/or processing algorithms, rather than low-level issues such as register usage or common subexpression elimination. The third major tool involves improving the programming environment available to users. This includes automatic correction of user errors, aids for debugging and testing user programs, mechanisms for maintaining the consistency of source and object programs, means for propagating changes throughout a program, and various other bookkeeping mechanisms related to the programming effort.

Middle Road.

The middle road is mainly concerned with exploiting a model that has been prewired into the system for use in a specialized area,

and with integrating existing technology from both artificial intelligence and the low road in construction of a coherent system. It is thus characterized by its deep involvement with a particular application area, its exploitation of a prespecified model, and its usage by nonprogrammers. These systems tend to be large, embodying thousands of facts and relations about the application area. They tend also to work in very narrow areas and utilize special processing techniques which enable them to optimize all or part of the processing specified within that domain. In performance they tend to exceed average human competence but are, as yet, unable to match expert ability.

High Road.

The third and most remote approach is the high road. Its basic concern is with the dynamic acquisition of models for the user's application. As such, its performance of tasks within the user's domain must rely on its ability to utilize domain-dependent knowledge of unknown form contained within these models. Thus, it is concerned with the space of possible models, with understanding the interactions between parts of these models, with the ability to recognize incompleteness and inconsistency within the models, and the ability to perceive and utilize causal relationships to achieve a goal within these models. It is also concerned with how these models can be communicated between the users or experts who understand the area and the system itself. Besides the technologies being developed in the low and middle road approaches, it is clear that the high road approach also requires major breakthroughs in artificial intelligence, modeling techniques, and communications technology.

Using the above conceptualization, the following categorization of ongoing work resulted. (See Figure 3.1.) The approaches are shown in terms of the high, medium and low roads versus the specificity of the system. The indentation in the medium approach indicates that it is, by its nature, an approach at specific systems or specific classes of systems.

Theoretical Approach.

It was mentioned almost in passing in the meetings that the theoretical level was perhaps orthogonal to the issues discussed above, i. e., correctness is necessary at all levels. It was also stated that the theoretical approach was getting much more practical via verification work, formal semantics, and theory of programming. In fact, practical verification can be seen as an extension of compiler technology. Certainly the case for the importance of the theoretical approach in automatic programming research was not clearly made and was a topic at the second meeting.

Problems Within Each Approach

The last major area covered was the problems which had to be solved for the different

	GENERAL	SPECIFIC	TYPICAL LANGUAGE
HIGH ROAD	ISI	Hacker	GPS
MEDIUM ROAD		MAC-Martin MATHLAB Heihorn IBM-Howe	PLANNER-like
LOW ROAD	Harvard	Self Strong	Lisp

Figure 3.1 Approaches to automatic programming.

approaches to attain an automatic programming system. As presented, the problems overlapped quite a bit and do not reflect the distinct areas of concern that seem to be apparent in the above conceptualization; therefore, they have been rearranged and modified to place them in this framework.

Low Road Problems.

The low road approach has the following problems:

1) the move from essentially macro-based languages to essentially context-dependent ones;

2) incorporation of understood concepts (e.g., error handling, flexibility, modification) into new or existing systems;

3) data representation and its interdependence with procedures;

4) the software factory, including both module selection and interface matching;

5) representation theory, including what the representation is really being used for, what is it accidentally being used for, and what pertubations can be made to it without affecting it's real use;

6) global program manipulation trade offs;

7) optimization, including the dropping of irrelevancies, representational abstraction, and computer-resource allocation.

Middle Road Problems.

For the middle road approach, the following problems exist:

1) the implementation, use, and evolution of asynchronous strategy-directed models, including the study of demon interactions and language for the domain-dependent and domain-independent strategies;

2) understanding the nature of the application area and finding ways to encase it in computable form.

Middle and High Road Problems.

The following are problems that relate to both the middle and high road approaches:

1) domain definition and handling of exceptions within the problem statement;

2) optimization at the model level;

3) model viewing;

4) responsibility assessments.

High Road Problems.

For the high road approach the problems are:

1) communications technology, including the ordering of information for presentation, the establishment and use of context, the effect of the recipient's capabilities on communication, and the perception of the recipient's capahilities hy the speaker;

2) modeling technology, including space of possible models, understanding the interaction between parts of a model, ability to recognize a model's incompleteness and inconsistencies, and the ahility to perceive and utilize causal relationships to achieve a goal.

Commonly-shared Problems.

There appear to be two problems that relate to all the levels. They are:

1) maintenance problems for long-life systems, including machinable documentation, modification of the system's function, and maintenance of the system's function as the environment changes;

2) movement between levels in terms of both debugging and documentation.

Theoretical Approach Problems.

Finally, there are a set of problems which deal with the theoretical approach. They are:

1) axiomization of complex data structures;

2) structuring of a program for verification;

3) more facts on low-level complexity theory;

4) the ability to prove numerical results within a range.

Report and Meeting Summation

The three distinct approaches to automatic programming form a nice covering of the field in that they are nested so the results of one should be available to those at a higher level. And although the low and high roads are largely fragmentary studies aimed at solving particular problems within those approaches, the middle course is primarily the practical systems approach which should push the methodologies on either side of it.

There were two main criticisms of the views expressed in the report [1]. First, it is too ambitious. The goal of domain-independent usage by those who are not professional programmers is unattainable within a time period appropriate for an ARPA project. Second, this emphasis and the way that the report was written limit the work that could go on under the other approaches which their proponents feel have a higher probability of success within an appropriate time frame.

LANGUAGE-INDEPENDENT PROGRAMMER INTERFACE

In view of the above concerns, we have very recently begun to explore a low road approach based on the creation of a facility for software production by professional programmers utilizing tools created on separate machines, but operating within a common framework via the ARPA Network. Such an approach has the advantage that some of the tools necessary for effective software production already exist, but not in a common environment where they can be effectively employed along with other tools. Further, such an approach is incremental: the system grows as new tools are created and added to the system. The tools available in such a facility would include code generators for a variety of machines, test data generators, regression analysis programs, on-line documentation aids, verification programs, etc.

The feasibility of such an approach has been greatly enhanced by our development of a language-independent programming environment. This work was conceived and implemented very recently. As such, the implications and significance of it are not yet clear, but the possibilities are very exciting. An interface has been built between two programming systems: BBN LISP and EL/1 (produced at Harvard). This interface allows the user to program in the EL/1 language, but with most of the facilities of BBN LISP as his programming environment. This environment has been extensively human-engineered and it undoubtedly is the best on-line programming environment yet created. These facilities include automatic spelling correction; a powerful program-structure editor; break, trace, and advise (modifications of the interface between one routine and another) capabilities; the ability to modify and/or reissue previous commands; and the ability to undo the effects

of previous commands, thus recovering earlier states.

Thus, the EL/1 user has a greatly expanded facility for creating, editing, and debugging programs. That programming language has been transformed into a programming system. The importance of such a transformation cannot be overstated in terms of programmer productivity. The significance of this work lies, however, not in the interfacing between LISP and EL/1, but rather in the observation that very little of the interface or the capabilities available in the BBN LISP programming environment are language dependent. Thus, once the system is operational and documentation has been produced, any language (with the three attributes described below) can be easily accomodated (a few man-days of effort is estimated). Furthermore, since all communication between the LISP environment and the programming language is via the ARPA Network, the language can run on any machine attached to the Network.

Three properties required of a programming language for use in this system are: 1) it is available over the ARPA Network; 2) it has an on-line evaluator (either an interpreter or fast compiler) and can field breaks or errors within a computation; and 3) either in such breaks or at the top level, it can evaluate arbitrary forms in that language.

The system is currently demonstratable, but many of the facilities are either unimplemented or undebugged. However, no problems are foreseen in completing the system. It should be operational and well documented by the beginning of June 1973. Figure 3.2 is an example of EL/1 usage.

From the standpoint of automatic programming, the significance of this work lies in its demonstration that the facilities required to produce software are largely language independent. This validates the feasibility of ARPA's approach of creating a system for software production with a common set of tools available for a wide variety of languages and capable of producing code for a wide variety of machines.

PROBLEM ACQUISITION

In support of the views expressed in the *Automatic Programming* report [1], the major research effort here has been in the area of *problem acquisition* which is concerned with obtaining an understanding of the user's problem and the domain in which it exists. This is necessary so that the following *process transformation* phase can attempt to find a sequence of transformations or operations in that domain which will obtain the solution required by the user. Thus, the problem acquisition phase is concerned with building the complete model of the user's domain; the model represents the interactions between the entities of that domain, and the effect on these entities by the allowed transformations or operations applicable within the domain. It is our primary contention that only through the development of such a complete model of the user's domain can the automatic programming system have any degree of generality in the domains for which it is applicable.

Currently, all such models of user domains have been coded into a system. We are proposing that such models can be specified to the system by its users, and that through these models the system can acquire the knowledge necessary to solve problems

```
->3+4;
NIL
->3+4$
(7)
->3+4
(7)
->TEST1_EXPR(A:INT,B:INT;INT)BEGIN A+B; END;
NIL
->TEST1(3,4)
(7)
->EDITF(TEST1)
EDIT
*PP
  (EXPR (A : INT , B : INT ; INT)
        (BEGIN (A + B)
               END))
*F BEGIN P
(BEGIN (A + B) END)
*(2 (A GT B => A-B; A+B))
*PP
  (BEGIN (A GT B => (A - B))
         (A + B)
         END)
*(-4 (A=B -> B_2*A))
*PP
  (BEGIN (A GT B => (A - B))
         (A + B)
         (A = B -> (B
                    2 * A))
         END)
*UNDO
(-4 --) UNDONE.
*PP
  (BEGIN (A GT B => (A - B))
         (A + B)
         END)
*USE -3 FOR -4
*PP
  (BEGIN (A GT B => (A - B))
         (A = B -> (B
                    2 * A))
         (A + B)
         END)
*OK
TEST1
->TEST1(3,4)
(7)
->TEST1(4,3)
(1)
->TEST1(4,4)
(12)
->ADVISE(TEST1 BEFORE (A_2*A))
TEST1
->TEST1(3,4)
(2)
-> USE 6 3 10 FOR 4
(18)
(3)
(16)
->ADVISE(TEST1 AFTER (VALUE\\_VALUE-1))
TEST1
->REDO USE
TYPE FAULT
- BROKEN
NIL
TYPE FAULT
- BROKEN
NIL
TYPE FAULT
- BROKEN
NIL
3:>RETBRK(0)
NIL
->TEST1(3,4)
TYPE FAULT
- BROKEN
NIL
1:>BT(1)
BLOCK: VALUE\\ _ VALUE - 1
TEST1
(NOTHING)
1:>EDITF(TEST1)
EDIT
*F VALUE 0 P
(VALUE\\ _ VALUE -1)
*(R VALUE VALUE\\)
*OK
TEST1
1:>TEST1(3,4)
(1)
1:>RLTBRK(0)
NIL
->REDO USE
(17)
(2)
(15)
```

Figure 3.2 An example of EL/1 usage. The EL/1 prompt character is either → or a number followed by a : followed by a > sign. Thus, user input will follow one of these symbols. The underscore represents a left assignment arrow.

within these domains and the understanding of what is required for such a solution. The two main issues, then, are what constitutes an adequate and appropriate model, and how such a model is specified or communicated to the system.

Adequate and Appropriate Model

A model is adequate if it contains a complete enough description of the relations between the entities in the problem domain so that a sequence of operations or transformations on this model can be built to solve the problem posed by the user. This is what was referred to as the complete model earlier. (See Problem-oriented Model above.) Thus, the adequacy of a model is dependent upon the use of that model to solve the problem. Operationally, this requires that the automatic programming system is capable of finding the complete set of applicable transformations on the model and calculating the consequences of each of these actions. The appropriateness of the model is a measure of how well suited the available transformations are to solving the problem at hand, i.e., an adequate model can be made more appropriate by adding to it nonprimitive transformations, made up of a sequence of primitive ones, which are suitable building blocks for the problem being posed. The model may also be made more appropriate by including recommendations about the suitability of alternative strategies for sequences of model transformations.

It is expected that the well-known problem of building a powerful general-purpose problem solver will be significantly aided by the user's tailoring of the specified model to make it more appropriate for the problem at hand.

Communication of the Model

The state of the art in natural language understanding is adequate for the description of problems and of models and is beyond our ability to utilize the information thus obtained, and hence, should not be a bottleneck in an automatic programming system. Evidence for this viewpoint comes from the work of Woods in the Moonrocks Program, from Winograd in the Blocks Description Program, and from Martin Kay in the Mind System. Each of these systems represents an alternative linguistic technology and each is capable of handling a wide range of linguistic forms within the domain of its competence.

The basic viewpoint then is to process the user's natural language communication with the understanding that it is meant to convey to the automatic programming system a model of his problem domain. Towards this end, the system can extract entities and the relationships between them from the communication. It can further query the user as to the relationships between entities which have not, as yet, been explicitly specified, but which have been inferred by the previous communication. Such inferences by the system about the incompleteness of the model require a sophisticated understanding not only of the communication, but of the types of models used for problem-domain specification. Unfortunately, sophistication in both these areas is quite limited. The need in communication is to be able to understand how information is ordered for presentation, how context is established and utilized, how the capabilities of the recipient effect the communication, and how these capabilities are perceived by the speaker. The need in modeling

is to have a space of possible models, an understanding of how the parts of a model interact, a means for recognizing incompleteness in models, as well as inconsistencies, a means for obtaining all the allowed operations on the model, and the means for transforming the models with these operations.

Human Use of World Knowledge

An experiment was devised to support our contention that only very limited use of world knowledge was necessary for problem acquisition in a new domain. Such an experiment requires the ability to cut off, or limit, the interactions between the new area being explained to the system and the body of world knowledge that might be brought to bear on understanding such a new environment. This could be most effectively done by translating each of the content words of the new domain into a nonsense word, and presenting the information as a mixture of normal English with the translated nonsense words appearing wherever one of the content words of the domain would have been used.

A number of such experiments were run, interactive and noninteractive, with both simple and complex problem domains, and the following conclusions were reached:

1) subjects are able to acquire knowledge about a domain described in unfamiliar terms;

2) the use of function words in such descriptions is very important to the understanding process;

3) some portion of such acquisition seems to be mechanizable through a set of rules;

4) subjects utilize world knowledge extensively to test local plausibility of interpretations;

5) style is an important aspect of a description which enables subjects to determine what are the important sections and how a description flows from one sentence to the next.

In addition, the protocols of these experiments have been analyzed and a set of rules obtained which, it is felt, will begin to define the way such models are built and knowledge is extracted from such passages.

Finally, a representation of knowledge is being prepared which would allow utilization of acquired knowledge in multiple ways, such as finding all elements which satisfy a certain relation, testing whether a particular element satisfies that relation, and deducing properties that elements which satisfy the relation must have. Such multiple use of information is critical in making maximal use of new items of information to gain an understanding of an environment. The proposed representation is halfway between procedural and table-driven representations of knowledge, such as resolution theorem-provers; it maintains, it is felt, the flexibility of the procedural representation while allowing the use of that information to make inferences on the relationships themselves, as in declarative table-driven type systems.

Efforts in this area are very formulative and it will be some time before practical results of this approach are available. The eventual impact, however, warrants such long term involvement.

REFERENCES

1 Balzer, R. M., *Automatic Programming*, USC/Information Sciences Institute, RR-73-1, September 1972, (draft).

PUBLICATIONS

1 Balzer, Robert M., *Automatic Programming*, USC/Information Sciences Institute, RR-73-1, September 1972, (draft).

2 ---, *CASAP: A Testbed for Program Flexibility*. USC/Information Sciences Institute, RR-73-5, February 1973.

3 ---, *Another Look at Program Organization*. USC/Information Sciences Institute, RR-73-6, February 1973.

4 ---, *Human Use of World Knowledge*, USC/Information Sciences Institute, RR-73-7, February 1973.

5 ---, *A Global View of Automatic Programming*, USC/Information Sciences Institute, RR-73-9, February 1973.

DOCTORAL THESES IN PROGRESS

1 Lingard, Robert W., "A Representation for Semantic Information within an Inference-making Computer Program".

2 Wilczynski, David, "Interactive Domain Acquisition for Automatic Programming Systems".

4 COMPUTER SOFTWARE ASSURANCE

Project Leader: **Ralph L. London**

Research Staff: **Richard L. Bisbey II**
Jim Carlstedt
Donald I. Good

Consultant: **Gerald J. Popek**

Research Support Staff: **Dorothy M. Fischer**

INTRODUCTION

This project, begun in September 1972, is developing methodologies, techniques, and standards for the analysis, testing, and evaluation of the software security features of operating systems. The project is providing a variety of means for answering the question, "Given an operating system, what tests and procedures can and should be applied in order to determine whether the operating system can withstand accidental or malicious attempts to access or destroy protected information, and how should it be designed so as to withstand such occurrences?". In other words, the research is aimed at directly supporting the software security requirements issued by DOD security policy-making agencies.

Project Approaches

Three separate but complimentary approaches are being pursued: *Empirical System Study*, *Protection Theory*, and *Program Verification*.

Empirical System Study.

Empirical system study focuses on near-term solutions to the requirement to operate computers, with classified and unclassified information simultaneously, in resource-sharing mode. Currently, DOD has over 3,400 computers in its inventory, with an additional 1,600 computers in use at its contractors' facilities, none of which can be operated in resource-sharing mode because none have an operating system which can withstand malicious attack. While there is considerable research in the production of secure operating systems, the results of this research will only be felt in the long term since even if secure systems were currently available, it would take at least five years for the government and its contractors to replace the current computer inventory and convert to the new systems. Faced with the requirement of operating in resource-sharing mode and the lack of operating systems to support such a requirement, this research is aimed at finding interim solutions by identifying:

1) vulnerabilities of contemporary systems and how these vulnerabilities transcend manufactures and architectures;

2) security trade offs which would allow contemporary systems to operate under restricted forms of resource-sharing; and

3) policies and procedures for testing and approving systems for such operation.

The work to date has resulted in the development of a new operational protocol for testing and evaluating operating systems, the categorization of known security flaws in contemporary systems into generic classes which

can be used to predict errors in future systems, and the successful use of the above protocol and categorization in discovering security flaws in the Multics operating system.

Protection Theory.

With respect to protection in operating systems, there exists much practice but little supporting theory. This subproject seeks to help remedy this situation by developing:

1) a basis for more formally describing *protection-forms* and *protection-mechanisms*;

2) a methodology for abstracting the protection-forms of given protection schemes; and

3) a framework for the comparison and evaluation of protection schemes themselves.

The inductive and deductive approaches employed have resulted in the beginnings of a taxonomic framework. Byproducts of this subproject are an annotated protection bibliography and a glossary of protection-scheme terminology and jargon.

Program Verification.

Verifying a computer program means demonstrating that the program is consistent with documentation or statements of what the program is to do. Current work is concentrating on providing computer assistance for verifying programs. One operational aid resulting from the group's work, called a *verification condition generator*, takes as input a program to be verified and also documentation of what that program is to do. The output from the verification condition generator is a set of mathematical lemmas, called *verification conditions*. Proving these lemmas demonstrates that the program is consistent with its documentation, and hence the program is verified.

Numerous nontrivial programs have been verified by proving by hand the lemmas. Several of these programs have also been verified by using an interactive theorem prover. The success of the verification condition generator has suggested additional components to increase computer assistance in verifying programs.

The empirical system study, besides obtaining certain results that apply to current operating systems, provides useful input to the other two areas. Both strengths and weaknesses of current operating systems must be accounted for in protection theory and in program verification. Protection theory provides some of the statements to be verified about secure systems. In turn, the empirical study is influenced by the other two areas.

Even though such matters as physical security, personnel security, communication security, user identification, hardware correctness, and hardware reliability are important and challenging parts of the overall security problem, they have been excluded from this work in order to concentrate on the software aspects of the overall problem.

EMPIRICAL SYSTEM STUDY

This subproject is characterizing contemporary operating systems in a security sense, identifying their strengths and vulnerabilities, and developing an empirical methodology for discovering security flaws in future systems. The work to date has resulted in the development of a new operational protocol for testing

and evaluating operating systems, the categorization of known security flaws in contemporary systems into generic classes which can be used to predict errors in future systems, and the successful use of the above protocol and categorization in discovering security flaws in the Multics operating system.

Operating System Characterization

Old Protocol.

All previous security test and evaluation (ST&E) activities known to this group have been carried out under war game-like conditions, with an aggressor trying to penetrate the system, and a defender attempting to detect such attempts. In general, the determination that a given system was either insecure or secure rested entirely with the success or failure of the penetration attempt [1-3].

War game test. An example of the extreme to which ST&E protocol has been carried is best exemplified in a recent test in which, in addition to finding a security flaw, the test was subject to the following conditions:

1) the security flaw had to be used in an undetected manner to repeatedly access a pseudo-classified file;

2) no information concerning the name and contents of the pseudo-classified file was given to the ST&E people;

3) no information concerning the [illegible] structure of the system being tested [illegible] given to the ST&E people prior to the test;

4) the entire test was to be conducted within a two week period.

The user, on the other hand, could know the names of the ST&E people and could actively defend the system by isolating and selectively scrutinizing the ST&E inputs, by unrealistically disguising the pseudo-classified file, and by making unannounced changes to the system during the test. It is unimportant whether these were, in fact, done; that these actions were not precluded is important.

While the ST&E people did meet all of the above conditions and thus showed the system to be insecure using the above definition, the activity was clearly not an analysis of the security of the system, but a test of the ingenuity of the ST&E people in implementing programs to manipulate the computer hardware and operating system after control had been usurped from the system. This was dramatized by the fact that over 75% of the ST&E activity was devoted to implementing programs to determine the security perimeter, to testing the security flaw, and to exploiting the flaw in finding and accessing protected information. Less than 25% of the ST&E activity was devoted to actually analyzing the security features of the system.

Problems. Major problems associated with the old ST&E protocol are the lack of stability of the system being tested as a result of system modifications made during the test, the use of penetration programs to verify the existence of security flaws, the limited range of possible test results of the ST&E activity, and the testing of systems under their best case conditions.

New Protocol.

The new ST&E protocol eliminates these problems.

Stable system. The first feature of the new protocol is that the test is carried out on a system whose development is frozen for the

duration of the test. As such, both the manufacturer and the installation are prohibited from making modifications to the system during the test. This does not preclude the improvement of the security features of the system during the ST&E activity, but it does preclude the incorporation of those improvements into the system being tested once the test activity has commenced. The importance of this test requirement is to prevent a reoccurrence of the following.

In a recent test, an installation, upon scrutinizing the ST&E input, saw that their system was about to be penetrated and proceeded to lock the ST&E people out of the system for a three week period while system modifications were made to fix the security flaw. At the conclusion of the test, the installation denied the existence of the original security flaw.

No programming. The second feature is the elimination of the writing of programs to verify security flaws. Instead, the verification is by joint agreement of the manufacturer (and/or installation) and the ST&E people. Only in the unlikely event of disagreement are programs written to verify the flaw. This eliminates situations such as the one cited above in which ST&E people devoted more time to exploiting a single security flaw than to analyzing the system for other flaws.

Expanded test results. The third feature involves the expansion of the range of possible outcomes of ST&E activity. The security determination of a system is far from settled by a penetration attempt. At least the following can result from a summary of ST&E activity:

1) the system was proven to be secure;

2) the system is believed to be secure;

3) the system is believed to be insecure;

4) the system was proven to be insecure;

5) no comment can be made about the security of the system.

An expanded scale such as the above, in addition to more adequately summarizing the ST&E activity, eliminates the anomalous situation in which penetration is equated with total insecurity while nonpenetration is equated with total security.

Worst case. A final feature is the evaluation of systems under worst case conditions. The goal of the ST&E activity is to provide DOD Security Policy with a full and impartial analysis of the system under test, and this can only be achieved with worst case testing. It cannot be achieved where war game conditions are imposed on the test, where the test must be carried out in a limited time, where all information concerning the system is withheld, or where extraneous activities such as finding a pseudo-classified file are required. A comprehensive analysis can only be carried out when the ST&E people have complete access to the system and all information about the system, and the ability to test under actual operations.

Security Flaws Handbook

Work has begun in generating an ST&E handbook. The handbook is intended for use by DOD personnel in evaluating operating systems. The handbook is written in two sections. The first section details the organizational aspects of security threats by presenting a methodology for establishing and operating a penetration effort.

The second section details specific security flaws found in various contemporary operating systems and discusses their generalization to other operating systems. This section is of particular importance to persons responsible for designing or evaluating systems since it not only provides an annotated list of security flaws, but also a framework for predicting where future flaws will most likely occur. Portions of this handbook have been presented at Department of Defense Computer Institute courses on computer security. (See ACTIVITIES below.)

New Protocol Test

Using the newly formulated ST&E protocol and the handbook, a test was conducted of the Multics operating system. The Multics system was chosen for the test because it represented the most advanced general-purpose operating system in which information protection was a fundamental objective of the design. The test resulted in the discovery of security flaws in the Multics system and demonstrated the practicability of both the protocol and the handbook.

Several interesting and beneficial results were noted. First, no machine time was used in the ST&E activity. All proposed security flaws were either verified or discarded with a single telephone call. As a result, there were no machine costs and no disruption of computer service to the Multics user. Second, other than the ST&E handbook, the most effective tool in discovering security flaws continues to be an information retrieval system for storing, retrieving, and operating on source-level documentation.

Research Trends

Research in empirical system study for the coming year will include:

1) The continued development of the ST&E handbook with the inclusion of results from other computer security studies, such as the Air Force Computer Security Technology Planning Study [4].

2) The continued dissemination of this project's successful methodology for discovering security flaws. This includes participation in Department of Defense Computer Institute and other governmental seminars in computer security, as well as nongovernmental lectures, seminars, and symposia.

3) The integration of this project's security flaw discovery methodology with the ARPA-sponsored MIT effort to develop a certifiably secure version of the Multics operating system.

4) The incorporation of the above test protocol into the ADP Security Manual [5].

PROTECTION THEORY

Protection: Control of Operations.

Protection in an operating system is concerned with the prevention of certain operations under certain conditions, where an operation is the application of an *operator* by a *subject* (active entity) to one or more *objects* (passive entities). Operators may range from primitive read and write accesses, through basic processor instructions, to software procedures of any size or degree of complexity. Subjects are users and the internal processes for which they are directly or indirectly responsible, as well as the processes of the operating system itself. Objects may be anything which can be denoted as operands, including

cells of virtual or physical storage and the values represented in them, composite information structures (including procedures), various other types of resources, and processes themselves.

Protection Schemes.

As parts of operating systems, protection schemes exist in various conceptual, design, implementation, and operational versions. A particular progression of versions, from conceptual image to operational system, characterized by increasing degrees of completeness, concreteness, and formality, is called a *version genealogy*.

In addition, the protection scheme specified by any version has both an external and an internal aspect, a distinction fundamental to the science of design [6]. The external aspect of a protection scheme, which exhibits its functionality from the point of view of the operating system's user environment, is called a *protection form* or *p-form*. The internal aspect, which consists of a set of protection-enforcing procedures and data bases, is referred to collectively as a *protection mechanism* or *p-mechanism*.

Problem Domain

This subproject has been underway for less than six months. Its original motivation came from the observation that for protection in operating systems, there exists much practice but little supporting theory: there are many schemes existing at various genealogical levels, but little attempt has been made to abstract, assess, or compare their respective p-forms, not to mention verifying p-mechanisms with reference to the p-forms they purport to fulfill. The goals of this subproject are to help remedy this situation by developing:

1) a basis for more formally describing p-forms and p-mechanisms;

2) the methodology for abstracting the p-forms of given protection schemes; and

3) a framework for the comparison and evaluation of p-forms themselves, both those associated with actual p-mechanisms as well as those for which no practical p-mechanisms have yet been proposed.

These interrelated goals have the following utilities:

1) A means for formally describing p-forms allows them to be more readily classified with respect to the kinds of protection they provide, i.e., it leads to a taxonomy of p-forms and thus of protection schemes themselves. A formal description language will also be of value in the future application of formal verification techniques to p-mechanism design and implementation versions. (See PROGRAM VERIFICATION below.)

2) The ability to classify, compare, and evaluate p-forms is of value both to designers who wish to know what kind of protection to provide in their systems, as well as to users who wish to know what kind of protection various systems can be expected to provide, and who wish to judge their appropriateness to particular social, commercial, military, etc. environments. A p-form taxonomy allows (or forces) operating system designers to declare their designs with respect to protection functionality, greatly facilitating these comparisons, evaluations, and judgments.

3) Progress in the development of a descriptive language and its basic terminology can only tend to enhance communication among designers and others working in the area of protection, and unify their efforts.

Complimentary Approaches

Both the inductive and deductive approaches are being employed. The inductive approach begins by surveying the existing population of protection schemes, as reported in the literature and known through firsthand contacts, and by studying selected representative schemes. There are two purposes:

1) to identify and distinguish generic characteristics versus peculiar ones, in order to form an empirical basis for description and model formation; and

2) to gain experience in abstracting p-forms.

The deductive approach is based on an analysis, quite independent of characteristics of actual p-mechanisms but within the theoretical context of operating systems, of the fundamental meanings of protection and consequent notions. This analysis is expected to result in a general p-form model, whose provisions encompass at least those of all foreseeably practical p-forms, and whose parameters are expressed in terms of their functionality. P-forms can then be described as instances generated from this model by specifying the parameters, which among other benefits, makes comparisons straightforward.

Employing the two approaches concurrently provides the opportunity for mutual validation of results. The deductive approach is also important because it allows the identification of classes of protection schemes which might be missed if only existing schemes were considered.

Reference Tools

Protection Bibliography.

The initial survey of protection schemes involved a review of the literature and the start of an annotated bibliography which is somewhat different from existing bibliographies [7-10] in that it focuses primarily on protection, to the exclusion of other aspects of security and privacy. The survey has been augmented by discussions with researchers from MIT/Project MAC, Carnegie Mellon University, and the University of California at Berkeley.

Protection Glossary.

Another result of this survey is a glossary of terminology and jargon used by various workers in connection with various protection schemes. Each entry in this glossary will contain references both to instances of its use as well as to other entries associated with the same or a similar concept. Such a glossary is felt to be a potentially valuable tool in reconciling similar and diverse efforts in the area, and is intended for publication.

Elements of a Protection Model

P-mechanism Elements.

From the initial survey the subproject has begun to learn how to identify, in any given protection scheme, its peculiar representation of those elements and that structure which are generic to all p-mechanisms. The essence of a protection scheme is a set of *permission functions*, the evaluations of which are preludes to various operations at hardware,

firmware, and software levels. Like an ordinary Boolean function, the inputs of a permission function are observable properties of selected entities; (functions of) these values are tested against reference values; a Boolean combination is applied to the resulting primitive propositions; and the output is its truth value. In the case of a permission function:

1) the entities whose properties are observed are normally one or more of the subjects, operators, and objects involved, although occasionally other values of the operating system state (e.g., current resource allocations) may be utilized; and

2) a result of "false" inhibits the operation in question.

The set of properties or state values utilized by permission functions are collectively called the *actual state* of the system. The collection of current permission functions and their associated reference values are the *policy state*. (See Figure 4.1.) A *protecting operator* is one to which a permission function is attached, and a *protected object* is one to which is applied only protecting operators which recognize the conditions associated with it. Both the actual state and the policy state must obviously be protected. It is clear that at least a part of the policy state must be variable, to reflect intents expressed by users with respect to protection of objects for which they are responsible, or to be created by the operating system on the basis of default policies when more explicit intent information is lacking [11]. It is equally clear that part of the policy state must remain constant, i.e., protected from any change, namely that part which protects the policy state itself at the highest metapolicy level. A framework for the comparative analysis of p-mechanisms is evolving from these and other observations.

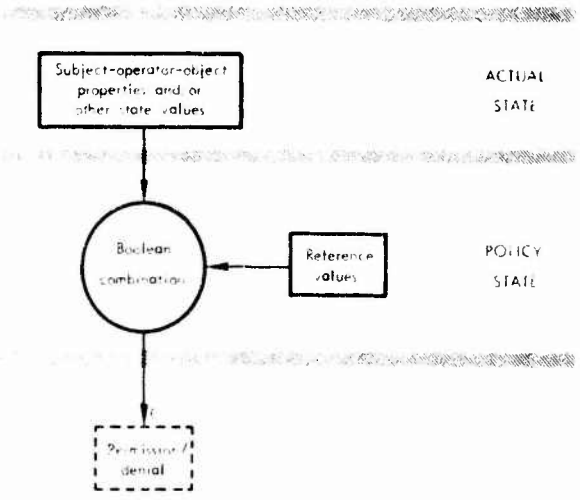

Figure 4.1 Permission function evaluation.

P-form Elements.

The study of fundamental implications (deductive approach) has been fruitful in that the basic elements of p-form description have been identified. The *functionality*, or behavior, of a p-form is defined in the following way. Its *scope* is the class of operations (i.e., subjects, operators, and objects) it covers. Its *conditionality* is the class of conditions which may be associated with these operations, expressed in terms of actual state properties and policy state values and conditionals. A p-form can then be regarded as a mapping from a class of user-expressed intents into some set of permission functions (with certain other permission functions fixed, as stated above), where intents as well as permission functions are

limited by the p-form scope and conditionality. Some of the obvious limitations on a practical p-form are that it must be implementable into a reasonably efficient p-mechanism, the scope can include only entities denotable by the user and identifiable by the operating system, and identifications must be nonforgeable. Any practical p-form is only an approximation to an ideal p-form in which any user intent whatsoever which can be stated informally can also he expressed formally and honored.

As a simple example of the usefulness of scope and conditionality as criteria, an important subclass of p-forms is singled out, namely those in which:

1) the scope includes only a few primitive operators such as "read", "write", and "execute"; and

2) protection conditions are regarded as relatively static.

This class is important because it has received the most attention and has been regarded as most widely applicable. In associated p-mechanisms, the results of initial permission function evaluations can generally be stored as *rights* or *capabilities* which are regarded as "possessed" by subjects, so that permission subsequently depends only on the test of a bit corresponding to the selected operator. This has led to p-mechanisms in which the protection and policy states are represented by sets of rights called *environments* or *domains* in which subjects operate and through which they move [12-14]. For more dynamic conditions, of course, such schemes are less appropriate, since when conditions change it may be difficult to revoke the dependent rights.

Research Trends

During the coming year the pursuit of the study of fundamental p-form elements and their relations will continue along with the analysis of p-mechanisms through a more intensive and thorough review of existing schemes and ideas. These two approaches are expected to produce a common description framework into which various p-forms and p-mechanisms can be conveniently placed.

In addition to the continued development of the protection theory glossary, it is planned to glean from the above-mentioned review a set of bench mark problems with which users and designers have been specifically concerned, to serve as one test of the generality of any p-form model which evolves. This is a large set; an attempt will be made to reduce it to a set of generic types by stating the problems in terms of the permission functions required for their solution.

The study of fundamental elements and relations has only begun, and several problems lie immediately ahead. The first is understanding how conditions of the policy state are attached to operators, subjects, and objects of protection. A related topic is that of *protection classes*: classes of subjects, operators, or objects whose protection conditions are the same. These play a prominent role in most existing schemes.

An important problem is the necessity of distinguishing among the various levels of representation which manifest themselves in operating systems. Scope entities are in many cases defined hierarchically. For example, a user is represented by the major processes he commands, these by sets of cooperating subprocesses or *tasks*, these in turn by sequences

of more elementary processes (of which the execution of a basic machine instruction is a primitive--but only with respect to the processor's external aspect). Operators exist in a similar representational hierarchy, as do elements of an information-data-storage structure. A p-form concerned with protection at a particular representational level cannot be directly compared with another which protects objects at a different level.

PROGRAM VERIFICATION

Verification--Definition, Problems, and Relevance to Protection

To verify a computer program means to demonstrate that the program is consistent with documentation or statements of what the program is to do. This demonstration is by rigorous, often formal, mathematical proof. Program verification is an important part of software assurance because the ultimate aim is to be able to prove statements about protection forms and protection mechanisms that are designed and implemented in operating systems. (See PROTECTION THEORY above.) The proofs of statements about protection, to be provided to users and managers of computer systems, will be part of the reasonable and adequate assurances that such systems do provide the desired level of protection. The significance to DOD becomes immediately apparent by substituting the word *security* for *protection*. A critical need is to permit computer access to classified documents while maintaining adequate safeguards against security violations.

Current verification techniques [15-17] are inadequate for proving protection statements about current operating systems. It is possible to prove statements about hundreds of lines of code, but not about thousands of lines, because of practical considerations of the amount of detailed, human effort involved. Moreover, personal experience has shown that verification is often made more difficult than necessary when programs are written without consideration of the need to verify the finished program.

These difficulties can be overcome or avoided. The MIT/Project MAC Multics group, among others, speaks of redesigning operating systems to provide a *security kernel* whereby the protection mechanisms are restricted to a hopefully small part of the entire operating system. Only the kernel would need to be verified, since all the protection is accomplished through the kernel; in particular, errors in nonkernel parts cannot affect protection. Whether the size and complexity of the kernel will permit verification is not yet known, but in any event the entire system need not be verified.

The amount of detailed effort currently required can be eliminated in several ways (to be explained below). The desire to verify programs is already having an effect on programming language design [18-20] with restricted features, syntax, and semantics being justified in part by this need. Numerous programming disciplines, such as chief programmer team [21] and structured programming (including few or no go-to statements) [22] are being advocated as ways to produce better programs. Obeying these disciplines will increase the likelihood that the resulting programs will be easier to verify.

There is evidence that programs involving parallel control structure can be verified [23-28] by using some of the same methods, sometimes generalized, that are successful on programs with only sequential control structure. The need to verify parallel programs arises, of course, since the protection mechanisms are part of operating systems which involve parallelism. Thus there is good reason to believe that statements on protection will be able to be verified at reasonable cost.

Existing and Planned Computer Assistance for Verification

Current subproject work, continuing joint work of S. Igarashi, R. London, and D. Luckham, is concentrating on providing computer assistance for verifying programs. The human effort required for verification is intended to be significantly reduced, but not totally eliminated since some human interaction seems essential as well as appropriate. The result will be an interactive system requiring key inputs, from a human, of the kind that would be considered natural for human understanding of what a program and its subparts do. Much or all of the necessary, but uninformative, details will be handled automatically.

The first part of such an interactive system is already operational in an experimental version as a Lisp program. This operational part, called a *verification condition generator* [29], takes as input a program to be verified and also documentation of what that program is to do. The output from the verification condition generator is a set of mathematical lemmas, called *verification conditions*. Proving these lemmas demonstrates that the program is consistent with its documentation, and hence the program is verified.

Input programs to the verification condition generator are written in Pascal [18], which is a precise, modern, Algol-like programming language. Acceptable input programs may contain assignment, while, conditional, and go-to statements; recursive procedure and function definitions and calls; and one-dimensional arrays. Thus a powerful subset of Pascal is processed. The documentation is provided by assertions in the form, roughly speaking, of quantified Algol Boolean expressions. The documentation is thus a slight extension of what programmers normally use to state conditions on computations that control their programs.

The theoretical basis of the verification condition generator is Hoare's axiomatic approach [30,31] for defining program language semantics. The semantics are given by means of proof rules for various program language constructs; by design, this facilitates the verification of programs. The semantics of Pascal are expressed in this way [32]. Formally, a verification is a demonstration of the desired result (consistent documentation) starting from appropriate mathematical lemmas and then using the proof rules. The verification condition generator, starting from the known desired result, continually computes subgoals (works backwards) until appropriate lemmas are computed. These lemmas are the verification conditions.

The verification condition generator is already a useful tool in verifying programs. Numerous examples of nontrivial programs, including published algorithms [33-35], have been verified by proving by hand the resulting verification conditions. Of greater interest

and usefulness is the fact that several arithmetic and array-sorting programs have also been verified by obtaining proofs of the verification conditions using an existing interactive theorem prover [36]. After human input of a small number of relevant facts, such as the associative laws and A + 0 = A, and after human selection of overall theorem-proving strategies, the theorem prover produced its proofs without further interaction. In one case involving sorting, the theorem-prover found an unplanted error in a hand proof, and, in effect, suggested how to correct it.

Future Research

Work on the verification condition generator is continuing in several directions which have been suggested by results to date. Changes will be made to eliminate the amount of detailed human effort currently required. A capability will be added to allow additional proof rules to be specified as part of the input. Additional features of Pascal, especially more data structures, will be allowed in acceptable input programs. The most pressing additional capability is to accommodate far less documentation and still obtain verification conditions that can be proved. Currently, the verification condition generator views the input program essentially as disjoint parts; the result is that the same documentation must often be separately specified for each part, an unnecessary duplication. Finally, much of what has been accomplished for Pascal programs is easily transferred to other languages. The verification condition generator may be extended to allow other languages as input.

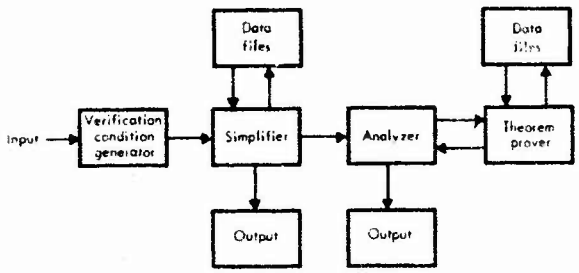

Figure 4.2 Proposed interactive system for verifying programs.

The success with the verification condition generator suggests additional parts of the interactive system for providing computer assistance in verifying programs. The overall plan is shown in Figure 4.2. Programs and documentation are input to the verification condition generator. Before attempting to prove the resulting verification conditions, it is often necessary to apply algebraic and logical simplification, and more generally, simplification using relevant properties of the operations in the input program. Such simplification is easy to do, and indeed, many of the verification conditions are proved by this operation alone. The verification conditions are very shallow mathematically with most being absolutely trivial and uninteresting. Hence, mammoth amounts of theorem-proving capability are not needed. Rather, from a human viewpoint, the tediousness and error-proneness of the proofs demand the assistance of the theorem prover. The analyzer is intended to reduce problems given to the theorem prover. It attempts to clarify verification conditions according to probable methods of proof and to generate simpler subproblems. It can also be used to find the "closest" similar condition that is provable when a proof of a given condition is not found, as a guide to human interaction. Since these approaches

have been used to obtain both hand proofs and proofs using the interactive theorem prover, the proposed system appears to have a good chance of being developed into something truly useful for software assurance. (Collaboration will continue with D. Luckham in this research.)

REFERENCES

1 Anderson, J., R. Bigsby, D. Hollingworth, and K. Uncapher, *Computer Security Experiment*, The Rand Corporation, WN-7275, March 1971.

2 *Final Report: Test and Evaluation of the McDonnell-Douglas IBM 370/155 Secured Operating System*, Lawrence Livermore Laboratory, University of California, January 1973.

3 Bisbey, R., R. Fredrickson, S. Glasman, D. Hollingworth, and W. Josephs, *Rand Test of the IBM DARWIN System*, The Rand Corporation, R-XXXX, (in progress 1973).

4 Anderson, J., *Computer Security Technology Planning Study*, Deputy for Command and Management Systems, Headquarters Electronic Systems Division (AFSC), L.G. Hanscom Field, Bedford, Massachusetts, October 1972.

5 *Techniques and Procedures for Implementing, Deactivating, Testing, and Evaluating--Secure Resource-Sharing ADP Systems*, DOD 5200.28-M, January 1973.

6 Simon, H. A., *The Sciences of the Artificial*, M.I.T. Press, Cambridge, Massachusetts, 1969.

7 Harrison, A., *The Problem of Privacy in the Computer Age: An Annotated Bibliography*, RM-5495-PR/RC, The Rand Corporation, December 1967, Vol. 2: RM-5495/1-PR/RC, The Rand Corporation, December 1969.

8 Hoffman, L. J., "Computers and Privacy: A Survey," *Computing Surveys*, Vol. 1, No. 2, June 1969, pp. 85-103.

9 Bergart, J. G., M. Denicoff, and D. K. Hsiao, *An Annotated and Cross-referenced Bibliography on Computer Security and Access Control in Computer Systems*, National Technical Information Service, AD 755 225, November 1972.

10 Anderson, R. E., and E. Fagerlund, "Privacy and the Computer: An Annotated Bibliography", Bibliography 30, *Computing Reviews*, Vol. 13, No. 11, November 1972, pp. 551-559.

11 Weissman, C., "Security Controls in the ADEPT-50 Time-sharing System", *AFIPS Conference Proceedings*, 1969 Fall Joint Computer Conference, Vol. 35, AFIPS Press, Montvale, New Jersey, 1969, pp. 119-133.

12 Dennis, J. B., and E. C. Van Horn, "Programming Semantics for Multiprogrammed Computations", *Communications of the ACM*, Vol. 9, No. 3, March 1966, pp. 143-155.

13 Lampson, B. W., "Dynamic Protection Structures", *AFIPS Conference Proceedings*, 1969 Fall Joint Computer Conference, Vol. 35, AFIPS Press, Montvale, New Jersey, 1969, pp. 27-38.

14 Graham, G. S., and P. J. Denning, "Protection--Principles and Practice", *AFIPS Conference Proceedings*, 1972 Spring Joint Computer Conference, Vol. 40, AFIPS Press, Montvale, New Jersey, 1972, pp. 417-429.

15 Elspas, B., K. N. Levitt, R. J. Waldinger, and A. Waksman, "An Assessment of Techniques for Proving Program Correctness", *Computing Surveys*, Vol. 4, No. 2, June 1972, pp. 97-147.

16 London, R. L., "The Current State of Proving Programs Correct", *Proceedings of the ACM 25th Annual Conference*, August 1972, pp. 39-46.

17 Linden, T. A., "A Summary of Progress Toward Proving Program Correctness", *AFIPS Conference Proceedings*, 1972 Fall Joint Computer Conference, Vol. 41, Pt. 1, AFIPS Press, Montvale, New Jersey, 1972, pp. 201-211.

18 Wirth, N., "The Programming Language Pascal", *Acta Informatica*, Vol. 1, No. 1, 1971, pp. 35-63.

19 Hoare, C. A. R., "A Note on the For Statement", *BIT*, Vol. 12, No. 3, 1972, pp. 334-341.

20 Morris, J. H., Jr., "Verification-oriented Language Design", *University of California-Berkeley Computer Science Technical Report No. 7*, December 1972.

21 Baker, F. T., "Chief Programmer Team Management of Production Programming", *IBM Systems Journal*, Vol. 11, No. 1, 1972, pp. 56-73.

22 Dijkstra, E. W., *Notes on Structured Programming*, Technical University of Eindhoven, The Netherlands, 1969.

23 Ballard, A., "Proving the Correctness of the SUE Timer Manager", 1972, (unpublished).

24 Habermann, A. N., "Synchronization of Communicating Processes", *Communications of the ACM*, Vol. 15, No. 3, March 1972, pp. 171-176.

25 Levitt, K. N., "The Application of Program-proving Techniques to the Verification of Synchronization Processes", *AFIPS Conference Proceedings*, 1972 Fall Joint Computer Conference, Vol. 41, Pt. 1, AFIPS Press, Montvale, New Jersey, 1972, pp. 33-47.

26 London, R. L., "Proof of 'Readers' and 'Writers' ", November 1971, (unpublished).

27 London, R. L., "Correctness of an Operating System Kernel", February 1972, (unpublished).

28 Parnas, D. L., "On a Solution to the Cigarette Smokers' Problem (Without Conditional Statements)", *Computer Science Report*, Carnegie-Mellon Univeristy, July 1972.

29 Igarashi, S., R. L. London, and D. C. Luckham, "Automatic Program Verification I: Logical Basis and Its Implementation", *Artificial Intelligence Memo 200*, Stanford University, May 1973; also USC/Information Sciences Institute, RR-73-11, May 1973.

30 Hoare, C. A. R., "An Axiomatic Basis for Computer Programming", *Communications of the ACM*, Vol. 12, No. 10, October 1969, pp. 576-580, 583.

31 Hoare, C. A. R., "Procedures and Parameters: An Axiomatic Approach", *Symposium on Semantics of Algorithmic Languages*, Engeler, E. (ed.), Springer-Verlag, 1971, pp. 102-116.

32 Hoare, C. A. R., and N. Wirth, "An Axiomatic Definition of the Programming Language Pascal", *Berichte der Fachgruppe Computer-Wissenschaften 6*, E. T. H., Zurich, November 1972.

33 Hoare, C. A. R., "Proof of a Program: FIND", *Communications of the ACM*, Vol. 14, No. 1, January 1971, pp. 39-45.

34 Floyd R. W., "Algorithm 245, TREESORT 3", *Communications of the ACM*, Vol. 7, No. 12, December 1964, pg. 701.

35 McCarthy, J., and J. A. Painter, "Correctness of a Compiler for Arithmetic Expressions", *Proceedings of a Symposium in Applied Mathematics: Mathematical Aspects of Computer Science*, Vol. 19, Schwartz, J. T. (ed.), American Mathematical Society, 1967, pp. 33-41.

36 Allen, J. R., and D. C. Luckham, "An Interactive Theorem-proving Program", *Machine Intelligence 5*, Meltzer, B. and D. Michie (eds.), Edinburgh University Press, 1970, pp. 321-336.

PUBLICATIONS

1 Bisbey, Richard L., Rod Fredrickson, Steve Glaseman, Dennis Hollingworth, and William Josephs, *Rand Test of the IBM DARWIN System*, The Rand Corporation, R-XXXX, (in progress 1973).

2 Igarashi, Shigeru, Ralph L. London, and David C. Luckham, "Automatic Program Verification I: Logical Basis and Its Implementation", *Artificial Intelligence Memo 200*, Stanford University, May 1973; also USC/Information Sciences Institute, RR-73-11, May 1973.

3 London, Ralph L., "The Current State of Proving Programs Correct", *Proceedings of the ACM 25th Annual Conference*, August 1972, pp. 39-46.

ACTIVITIES

1 Bisbey, Richard L., "Operating System Security", at Computer Systems Security Symposium, Department of Defense Computer Institute, Washington, D. C., August 29-31, 1972; November 14-15, 1972; May 16,

1973; IBM Scientific Center, Los Angeles, April 4, 1973.

2 London, Ralph L., "The Current State of Proving Programs Correct", ACM 25th Annual Conference, Boston, Massachusetts, August 1972.

3 London, Ralph L., member of the working group on Audit at the NBS/ACM Workshop on Controlled Accessibility, Rancho Santa Fe, California, December 1972.

4 London, Ralph L., "On Proving Assertions about Secure Systems", Sixth Hawaii International Conference on System Sciences, January 1973.

5 London, Ralph L., "Interactive Program Verification: A Logical System and its Implementation", Computer Science Conference, Columbus, Ohio, February 1973; University of California at Irvine, February 6, 1973; IBM Research Laboratory, Yorktown Heights, N.Y., February 23, 1973; California Institute of Technology, April 18, 1973; and University of California at Los Angeles, May 3, 1973.

5 NETWORK-SUPPORTING RESEARCH

Project Leader: **Thomas O. Ellis**

Research Staff: **Stuart C. Feigin**
Louis Gallenson
John F. Heafner
Thomas N. Hibbard
John T. Melvin
Robert H. Parker
Leroy C. Richardson
Guner S. Robinson

Research Support Staff: **R. Jacque Bruninga**
George W. Dietrich
Dorothy M. Fischer
Oralio E. Garza

INTRODUCTION

The ARPANET has clearly demonstrated the effective sharing of distributed computer resources. Yet the principles exhibited in its design have broad potential for improving the Network and other national and worldwide communications systems in versatility, reliability, survivability, and cost-effectiveness. This group has participated in the research and development of new ARPANET capabilities, as well as extending the earlier work of other ARPA contractors.

Project Elements

The Network program comprised the following elements:

- Network Conferencing
- Transparent Network Communications
- DOD Communications Study
- Accounting System Study
- Data Reconfiguration Service
- Portable Terminals

Network Conferencing.

The main thrust of ARPANET capability explorations to date has been in the area of computer to computer data/program communication. While this field is developing important new concepts in data, program, and other resource sharing, other modes of communication are necessary for the general needs of existing and future telecommunications systems. Specifically, the transmission of time-dependent, continuous, natural information (such as speech) represents a problem area for which solutions have not been demonstrated.

A study has begun here of some of the problems associated with speech communication via the ARPANET, along with the development of demonstrable solutions. Vocoding algorithms (developed by other ARPA contractors) and appropriate signal-processing hardware will be used by ISI to explore the Network voice parameters in a prototype system on the ARPANET. The signal-processing hardware will also be used to simulate the characteristics of various types of networks. Conferencing capabilities will later be extended to include the exchange of natural pictures (such as scenes) and computer-derived information.

As an interim measure (until high-speed signal-processing equipment can be procured), a network simulator has been written for an in-house STANDARD Computer Corporation IC-4000 computer. D/A and A/D converters have also been attached to it. With this configuration, signal stream segmentation is presently being studied.

Transparent Network Communications.

The overt ARPANET communication instructions that must be supplied in many uses of the Network tend to handicap the inexperienced user. Transparent networking attempts to suppress ARPANET protocols from the point of view of the user's programs. The means of achieving remote subroutine access and parameter passing are being studied. The goal is to determine the requirements of certain classes of remote process communications, and then demonstrate the techniques developed as a result of this determination. At the moment, various interprocess communications are being identified and classified in preparation for demonstration programs.

DOD Communications Study.

A study was initiated, in response to a DOD-expressed need, for alternative approaches to the consolidation of all military message communications on the island of Oahu, Hawaii. The basic goals were to improve service and lower costs using advanced telecommunications concepts. The proposed communications system plan offers both a system architecture and a working methodology to consolidate, through automation of the writer to reader service, the Island's message needs.

Accounting System Study.

A quick response study drafted an ARPANET accounting system architecture to provide Network-wide audit and accounting capabilities. The study's report identified resources to be accounted for and measurements to be recorded, and proposed an accounting system design and implementation milestones. An ARPA-sponsored group is now investigating, and will recommend, an accounting policy.

Data Reconfiguration Service.

The Data Reconfiguration Service (DRS), an experiment allowing communication between two arbitrary but cooperating remote processes, has been completed by ISI, the University of California at Santa Barbara (UCSB), and The Rand Corporation. The DRS permits a user to specify any desired data transformations between his remote processes in order for each process to receive data in its expected format. The two processes then converse as if they were directly connected. The DRS, however, monitors their dialogue and performs the user-specified transformations on data passing between them. ISI has concluded debugging the DRS compiler and testing the service. DRS is now offered, on an experimental basis, on the IBM 360/75 at UCSB.

Portable Terminals.

A portable terminal for use with the ARPANET is presently being developed. Most of the capabilities of conventional portable display terminals will be available, but this prototype will be both smaller and lighter than existing models. The case will fit under an airplane seat and weigh approximately one

third less than current portables. The terminal, using an acoustic coupler, will be connected to any standard telephone handset and will communicate from there to any ARPANET site. Completion date for the prototype is July 1, 1973. If this model is successful, the second implementation will have an increased character display, an optional graphics capability, and be even smaller in weight and size.

NETWORK CONFERENCING

This subproject is investigating the problems of multiple site intercommunications through a network such as the ARPANET. The current research scope includes the feasibility of secure, high quality, low bit rate, digital voice communications via a packet-switched network. In addition, human-factors considerations involved with communicating via this medium will be investigated. Image transmission, conference protocol, interaction of conferencing requirements, and computer/data file support will be explored later.

Network Conferencing Tasks

The tasks of the network conferencing system are:

1) To establish a wide bandwidth full-duplex, high quality, real time, voice link: this will be done first with the IC-4000 simulating the ARPANET. This equipment permits study of the human-factors elements of continuous signal stream segmentation over a store-and-forward, packet-switched network.

2) To implement a Pulse Code Modulation (PCM) voice link on the IC-4000. Speech will be sampled at a rate of 8,000 times per second, and then quantized using 6, 7, or 8 bits. This results in a bit rate of 48, 56, or 64 kilobits per second. A nonlinear quantization scheme will be used to achieve the best overall signal-to-noise ratio without increasing the sampling frequency and/or the number of quantizing levels.

3) To implement nonreal-time experiments with existing Linear Predictive Coding (LPC) bandwidth reduction techniques on the IC-4000. This will be done to gain an understanding of both the computation requirements for real time, low bandwidth LPC, and the effects of LPC on speaker identity and intelligibility.

4) To implement a signal processor system capable of simulating the ARPANET, as well as other digital networks, while processing LPC algorithms for low bandwidth, real time, full-duplex, voice communications.

5) To implement an ARPANET simulator and LPC algorithms on the signal processor. Continuous, real time speech will be segmented as required by the asynchronous nature of the ARPANET, or other digital networks. Once bandwidth has been reduced, methods of assuring continuity and natural-sounding speech will be studied. These experiments will buffer speech samples to assure continuity of speech over network dropout intervals and over different path-dependent transmission delays.

6) To implement multiple-site voice communications when practical network bandwidth has been achieved. It is assumed that another processor will exist on the ARPANET and that algorithms implemented on the ISI signal processor can be exported for dual site experimentation.

7) To test conferencing situations, including working distributed office communications, small technical conferences, and decision sessions (such as with the Delphi method).

8) To examine the broader conferencing problem involved in multiple media communications, such as slow frame TV or snapshot imaging, and remote hard copy or facsimile. A terminal will be developed incorporating these multimedia communications devices which expand shared information space. Additional scenarios will be generated as these tools are added to the terminal.

Experimental Configuration

A wide bandwidth, full-duplex, voice channel to a STANDARD Computer Corporation IC-4000 microprogram computer has been implemented. This implementation was selected primarily because of the availability of the in-house IC-4000 processor on a dedicated basis. The configuration will be used to simulate ARPANET message delays and their effects on continuous voice communication. The channel bandwidth is 5kHz. (about 25% wider than the standard telephone bandwidth of 4kHz.), and no bandwidth compression techniques are used. The system employs analog to digital and digital to analog converters with 9 bit resolution and a 100 μsec conversion rate. The total digital bit rate is, therefore, 90K bits.

The experimental configuration involves two sound booths, each with a high quality speaker and microphone (to maintain control over quality artifacts); this setup allows the experimenter to communicate as he would over a full duplex speaker phone. The experimenter's voice is amplified from the microphone, filtered by a 6 pole, low pass filter at 5kHz., and passed through the A/D converter to the IC-4000. Similarly, the second experimenter's voice is amplified, filtered, and passed into the IC-4000. The IC-4000 then simulates the delays which would be encountered if the speakers were conversing over the ARPANET. After the proper delay time, the IC-4000 sends the speaker's digitized voice to the opposite experimenter's D/A converter, then through another filter, the power amplifiers, and the speakers in the sound booths.

ARPANET Simulation

The ARPANET audio simulator operates under control of an IC-4000 microprogram. The microprogram emulates an IBM 7044 computer with an extended instruction set. An extended version of IBM's 7044 IBSYS provides the basic target machine software support.

The simulator can be driven by two audio channels. The audio signal is read and written at the rate of 36 bits per channel every 400 microseconds, and synchronized by interrupts from the analog-digital interface. Due to the limited speed capability of the IC-4000, the interrupt service routine, although carefully coded, uses about 65% of the CPU cycles. The remaining time, while not enough to allow extensive processing on a word by word basis, permits simulation of Network delays. The data is packetized, and both data and ready-for-next-message (RFNM) transmission delays are appropriately inserted in each channel. Network delays are calculated according to formulas reported by McQuillan et al. [1]. Up to 5 seconds of speech can be buffered by the IC-4000 for each channel. In addition, some periods of silence can be detected and

eliminated, thus reducing the average bandwidth. In the absence of ARPANET delays, speech quality is better than that on the telephone, and full-duplex operation has been achieved.

The Network simulation code has been implemented as have the hardware A/D and D/A converters. The LPC speech analysis and speech segmentation are planned for the future.

Vocoders and Vocoding Techniques

A field study was conducted to determine the state of the art of vocoders and vocoding techniques. Five possible sources of vocoders were found:

1) Sylvania markets a minicomputer, the Programmable Signal Processor, capable of Fast Fourier Transforms (FFT) and currently programmed for real time Cepstrum vocoding at 2.4 kilobits. Intelligibility is about 95%, and the cost for a two site system, with availability six months, would be $80,000-$150,000.

2) LTV Electrosystems, Inc. offers a channel vocoder that operates at 2.4 or 4.8 kilobits and uses a regular telephone handset. The availability of such a vocoder is three months and the system cost is about $17,000 per site.

3) Magnavox has a 2.4 to 4.8 kilobaud modem for attachment to a discrete FFT vocoder. A prototype is available and a two site system would cost $28,000.

4) Philco Ford offers a voice-excited, analog vocoder that operates at 9.6 kilobaud. It costs approximately $20,000 per unit and intelligibility is about 10-20% better than channel vocoders. They also have a Continuous Variable Slope Delta Modulation Technique at 10 kilobaud that exhibits good recognition with a 26-30 decibel signal-to-noise ratio. An analog implementation is available now; a digital version will be available soon. System cost is approximately $2,000 per site.

5) Lincoln Labs has developed a digital pitch vocoder at 2.4 kilobits and a rack-mounted voice-excited vocoder at 4.8 to 9.6 kilobits. These could possibly be loaned to ISI for experimentation.

6) Several sites* are studying LPC techniques for speech understanding but are at least a year away from 2.4 kilobits, real time, high quality vocoders.

State of the Art

The field study on vocoders and associated techniques produced the following conclusions which will be used in the project's work when a signal processor is obtained:

- No high quality hardware vocoder exists at 2.4 kilobaud or lower.
- No real time, high quality software vocoder exists for the same bandwidth.
- Because of the poor speech quality of the available vocoders, it would be unwise to procure a hardware vocoder for ARPANET experiments at this time.
- A signal processor would allow much more flexibility in algorithm manipulation and network simulation than would a hardware vocoder.
- A software vocoder allows any implemented algorithms to be shared among ARPANET conferencing experimenters.

---

* Bolt Beranek and Newman Inc., University of Utah, Southern Methodist University, National Security Agency, Speech Communications Research Laboratory, Inc.

• LPC is most likely to produce high quality speech at reasonable bandwidth, although present algorithms are not computationally practical.

• Project personnel will implement LPC algorithms, as they are developed by the speech research community. The signal processor necessary for vocoding algorithm implementation will also support network simulation for controlled study of Network effects under various conditions.

Initial Study of LPC of Speech

A preliminary study was made to determine the computational requirements for a signal processor which is capable of providing and maintaining high quality, full-duplex voice communications over the ARPANET. Current efforts are directed towards the Linear Predictive Coding (LPC) speech processing algorithms.

The theory of various forms of linear prediction has been treated in detail in the literature [2-16]. Several formulations of the technique are closely related but they have important differences. One aspect common to all formulations is that, at a particular time, a speech sample, s(nT), can be approximated by a weighted sum of the past p samples as

$$s(nT) \cong \sum_{k=1}^{p} a_k s(nT-kT)$$

where T is the sampling interval, p is an integer between 10 and 15, and a_ks are coefficients closely related to the glottal excitation function and the vocal tract transfer function of the human speech mechanism. The weights, a_ks, referred to as the predictor coefficients, are calculated to minimize the average energy in the error signal (which represents the difference between the actual and predicted speech amplitude). The coefficients are calculated over short speech segments of 10-30 milliseconds, and thus change as the speech statistics vary.

The error signal can be characterized by its total energy and a time distribution of either periodic pulses at the pitch rate or white noise, depending on whether the speech is voiced or unvoiced. Thus, in an LPC system, the error signal is not transmitted as in adaptive predictive coding [2], but, rather, its time and energy characteristics are transmitted.

LPC Speech Processing System

An LPC speech processing system consists of an analyzer at the transmitting site, and a synthesizer at the receiving site. At the transmitting site, short segments of speech are analyzed, i.e., the predictor coefficients are computed, the pitch is extracted, and the voiced/unvoiced decision is made either on the speech or error signal. At the receiving site, a reconstruction filter is formed (excited by the error signal) and gives an approximation of the speech signal as its output. Thus, the synthesizer need only know the predictor coefficients and gain factor, the vocal pitch, and a voiced/unvoiced decision parameter, in order to generate an approximation of the error signal which excites the reconstruction filter.

Various forms of LPC speech processing systems have evolved which differ in the method of computation of the predictor coefficients, the method of pitch extraction, and the voiced/unvoiced decision. Another important factor is the transmission parameters. Transmission of the predictor coefficients requires at least 8 to 10 bits per coefficient in

order to insure the stability of the reconstruction filter [3]. Therefore, transmission of an equivalent set of parameters must be considered. This transmission can be done by partial correlation (PARCOR) coefficients, area functions, or frequencies and bandwidths of the poles of the reconstruction filter. The set of PARCOR coefficients is being considered here. At the receiver site, speech samples are obtained directly from these parameters or via the predictor coefficients.

LPC Speech Processing System Information Rate

The information rate of the LPC speech processing system is given by

$$R=(N_p + N_r + N_v + N_g)/T_f \text{ (bits/second)}$$

where

$$N_p = \sum_{i=1}^{p} N_i$$

is the number of bits assigned to the predictor coefficients or any other related parameters (such as PARCOR coefficients). N_r is the number of bits for the pitch period; N_v and N_g are the number of bits for the voiced/unvoiced decision and the gain factor, respectively. T_f is the frame period to readjust all the parameters and $F_f = 1/T_f$ is the frame rate (in frames/second). For example, if a total of 72 bits are used for transmitting all the parameters, and if the frame period is 10 milliseconds (i.e., 100 frames per second), then the resulting bit rate is R = 7,200 bits/second. In order to achieve speech transmission at 2.4 kilobits, all the parameters represented by 72 bits must be readjusted every 30 milliseconds.

Experiments have shown that the following bit assignment is satisfactory for various control parameters of the LPC synthesizer: N_p = 60, N_r = 6, N_v = 1, and N_g = 5, resulting in a total of 72 bits.

Computational Requirements

Computation and storage requirements for a linear predictive speech coding system depend on the particular scheme used in computing the predictor coefficients, and the method of pitch extraction and voicing decision, among other factors. If the coefficients are obtained by covariance techniques [3], and sophisticated pitch extraction algorithms are used, then the total number of computations (a computation is defined as one multiplication and one addition) could be as high as 9,225 for each frame [16]. For a bit rate of 2.4 kilobits/second, if the frame length is 30 milliseconds (see LPC Speech Processing System Information Rate above), then the total number of computations is 307,500 per second; this implies that each computation must be performed in 3.25 microseconds. Accuracy studies indicate that such a speech processor will maintain high quality voice if calculations are performed with a floating point precision of a 16 bit mantissa and an 8 bit exponent.

Various researchers are presently investigating the possibility of successful operation of an LPC speech system with a fixed point processor. Reducing the total number of computations is also being investigated [17]. These techniques will be studied and some will be implemented on the signal processor, when the latter is obtained.

TRANSPARENT NETWORK COMMUNICATIONS

The ARPANET has brought to the fore the general problems of communication among programs residing on different machines. This

subproject investigates means for calling programs over the ARPANET (i.e., from outside of their host machine environment) in a manner transparent to the user. For example, a user of a list processing system might need an arithmetic computation capability better suited for another language. Similarly, a program could provide access to specialized data bases as well as to special operations on that base. In these instances, it is desirable for one program to invoke another as a subroutine. In principle, the programs need not reside on the same host; thus they may be distributed among several machines on the ARPANET.

Transparent Networking Tasks

Project tasks are:

1) To determine the requirements for programs that provide transparency for program to program coupling. Such requirements are of two kinds:

   a) language-independent considerations (e.g., the protocol between programs for passing parameters and other data); and

   b) language-dependent considerations involved in implementations for particular language classes (e.g., Lisp and PL/I).

2) To develop example programs that demonstrate the utility of the proposed techniques.

The requirements are being examined in terms of the conversion and transmission of data among programs residing on different machines, and the simulation of programs' address spaces with regard to passing parameters between them. A literature review of techniques currently used for data conversion and transmission (especially those used on the ARPANET) is being conducted. A systems analysis of the requirements imposed by the Network upon transparent program interactions is also being made.

When the requirements are ascertained, program to program communication for several systems will be implemented.

Data Conversion and Transmission

Methods for dealing with the problem of data conversion and transmission are:

1) use a common facility, such as the Data Reconfiguration Service (DRS), to accomplish the necessary conversions and transmissions;

2) implement particular protocols for transmitting data between any two machines;

3) use some common, data-transfer protocol for expressing all data in an intermediate, standard form, where each host is responsible for just the transformations to/from this intermediate form.

The DRS (see that section below) is currently being offered as an experimental facility at the University of California at Santa Barbara [18]. While this is a possible interim solution, it is not viable in the long run for performance reasons unless implemented in firmware.

The second method is the most common technique, with two sites using an agreed upon nonstandard scheme.

The third procedure is considered a substantial part of a viable transparent interprogram communication system. It avoids the availability and delay problems inherent in

using a third-party system for data conversion, and does not suffer from the incompatibilities and multiplicity of effort prevalent in the second method.

Transnetwork Parameter Passing and Address Simulation

The problem of transparent communication through address space mapping has not been examined except in cases of programs communicating through a shared (actual) address space [19,20] on a single machine.

Briefly, the problem is to determine that a program, A, has at some point attempted to access a variable that exists in the address space of another program, B, and thus access to that space must occur (e.g., a variable called by name). Upon recognizing the variable-name reference, the appropriate access must be effected. Program B need not be either directly used or invoked by program A; i.e., B could have called C which in turn called A, so that B's actual parameters have filtered through to A by way of C.

The problem can be partitioned as follows: 1) determine that a transparent network program call, or data transfer pursuant to a call (i.e., passing of parameter values, etc.) is needed; 2) establish connections to effect the call or transfer; 3) invoke the procedures for effecting the call or transfer.

DOD COMMUNICATIONS STUDY

This subproject is an integral part of ARPA's and ISI's program to explore the utilizations of computers and communications resources applicable to military environments. The task may be characterized as assisting the operational arms of DOD in capitalizing on its research products provided by ARPA. Within this framework, the specific goal of this study was to introduce an advanced teleprocessing architecture into a DOD operational environment to substantially reduce message-handling costs and delays.

A sizable complex, namely the DOD communications facilities and methods on the island of Oahu, Hawaii, was examined. Careful consideration was given both to the current operations on Oahu and the projected requirements. This data gathering was followed by a report [21] recommending a telecommunications system architecture and philosophy of operation.

The suggested communications and computer components are production items within a customized configuration suited to present needs. The architecture allows for addition or deletion of components to accommodate future requirements. The use of off-the-shelf units would allow the system to be operational within two years at a cost of approximately $25 million.

Briefings on the conclusions and recommendations in the report were provided for military communications officers responsible for the Pacific area.

Current Situation

To assure an adequate level of understanding of need, ISI staff members visited Hawaii in March and April, conducted interviews with DOD staff both in Oahu and in Washington, D.C., and used the results of data obtained from the Commander in Chief, Pacific (CINCPAC).

Military communications on the island of Oahu are handled by 2,000 communications

personnel trained in the handling and distribution of messages. They currently provide the interface between the people who wish to send and receive messages and the local communications equipment. The cost of this service is about $20 million per year.

Though needs of the various organizations and sites on the Island are reasonably common, limitations of the manual system require that message functions frequently be duplicated. Furthermore, within each organization, several stages of manual message accounting, copying, and courier-forwarding exist between the electronic communications and the action officer. For most action officers there is no secure communication means, other than hand routing, to coordinate messages either for preparation or for action.

This manual handling results in high communications costs, great inconvenience, and often very long delays in message preparation and delivery. For these reasons, the current message service is largely treated as a mail service, rather than a prompt message means. To lower costs and delays, the number of communications personnel must be reduced by automation of their accounting and routing functions, and by providing direct communications capability to the offices of message senders/receivers.

ARPANET Architecture

The particular system architecture proposed in the report is based on operational, state of the art, message-handling components. The recommended ARPANET technology is currently used productively by project administrators, coordinators, technical managers, liaison personnel, and secretaries, for their daily communication needs.

The ARPANET architecture was suggested because several hundred man-years of R & D have been invested in its various components. Very little additional effort would be required to customize this architecture for military needs and environments. Technically, this entire system could be operational in less than 24 months.

Costs.

The system can be installed for an estimated investment of: 20 to 40 man-years and a few million dollars for system component modification; approximately $25 million in capital costs; and $200,000 per year for leased line costs. It would provide Oahu-wide, consolidated, direct writer to reader automation, which would result in large savings, a significant increase in the efficiency of action officers, and a vast improvement in message promptness. However, if a new system is designed from the beginning, costs in excess of $50 million can easily be expected, with a final system delivered in three to five years.

Technology and Methodology.

It was determined that not only a technology, but also a methodology was needed to promote a smooth transition from the present manual arrangement to a largely automated, reliable, and cost effective system. The report suggests a well-proven system architecture and a functional methodology; it is emphatically felt that the two are inseparable if an effective, user-oriented system is to be constructed.

Expandable System.

Although the principal focus of the study was the solution of the Oahu communications problem, the system should, at the same time,

be viewed in larger context. Since the proposed communications architecture is relatively independent of physical area and of number of users served, the Oahu solution might serve as a prototype for other similar sites; it can also serve in expanded form as a worldwide military communications system.

The basic system architecture can accommodate new services and easily expand or contract without disruption of ongoing operations. Thus the Oahu military community will not be constrained by a specific mechanism for their current mode and level of operation.

On-line System

To fully realize the cost and functional benefits of an automated system, it is essential that on-line terminal communication be brought directly to the action officer and/or his secretary. This will not require a terminal for every officer on the Island, since in most cases several officers and support personnel are clustered in a single physical office space. Their traffic levels are such that several officers (average of 3) can share a terminal without conflict. By allowing these people to be directly on-line, the system can provide: complete accountability of messages; preparation aids; scanning aids; interoffice communication; message status reporting; task listing by priority, with audible and visual alarms, and many other personnel aids to enhance the effectiveness of action officers.

Bringing the communications directly to the user offers a far reaching set of capabilities that extend beyond anything currently available and with more complete security protection. Not only are delay times of only seconds attainable, but the possibility for informal interterminal communications can improve the operation of a unit and greatly improve security.

TENEX and Multics.

Two systems, TENEX and Multics, have already been developed and made operational. These systems, in essence, perform this kind of message service today. The TENEX system, based on the Digital Equipment Corporation (DEC) PDP-10, provides a message-handling service on the ARPANET; Multics, based on a Honeywell 6180, provides a capability for handling messages and for multilevel security safeguards. Either of these systems is capable of effectively supporting the communications requirements for Oahu. Software development is needed to tailor the system to the specific requirements of the users of Oahu. However, fundamental changes to the underlying operating system structure are not required.

Packet-switching.

A reliable communications system is needed to interface the users to these machines. The interconnection should be such that the users are totally unaffected by the failure of an individual computer and its location. The ARPANET packet-switching technology is extremely well matched to this kind of performance. Although other forms of communications can serve to interconnect the user and computers, the packet-switching application has been focused on in this plan because it is more reliable and more flexible in serving a wide variety of users and user needs. Further, the system is easily scalable in size.

General Services Network.

Communication of separate intelligence community traffic through a general services network is easily achieved. The network architecture proposed is flexible enough to extend to any capacities that can be reasonably foreseen in the near future. Further, the proposed system permits the interconnection of computers to provide other services. Also, the network can be easily interfaced to other communication systems (i.e., AUTODIN) for worldwide connectivity.

The Report

The report develops this plan in greater detail. It describes the communications problem on the island of Oahu, discusses the nature of the user requirements and shows how desirable and workable is the objective of placing the user directly on-line, proposes an overall system and describes its operation, covers the magnitude of the system required, and describes detailed cost and implementation considerations.

Briefly, then, the report describes a plan for consolidation of communications services on Oahu which relies on placing the users directly in contact with an automated message-handling service. This service would be provided by a set of computer-based message processors configured to provide message handling with security capability and high reliability. These processors would be interconnected with each other and the users via a packet-switching network. The entire system would provide essentially "immediate (seconds) message delivery". This approach is an effective, expandable way to proceed in consolidation of the telecommunications services on Oahu.

The recommendations cited in the report are currently under consideration by the Joint Chiefs of Staff.

ACCOUNTING SYSTEM STUDY

A short study was conducted to examine the accountability of use of the ARPA communications network and the extended network of ARPA computing facilities coupled by the communications network. An architecture and implementation plan for network accounting were drafted as a result of the examination. The study was initiated because of the recognized need for an automated accounting system. However, discussions during the study of accounting and management needs revealed that an accounting system is only the kernel of a broader, partially-automated ARPA resource management system.

Basic, Open-ended System.

A basic accounting system that can be made operational on a much shorter time scale than a resource management system is needed. Thus, the substance of the resulting report is a model for an accounting tool that stresses reliability, yet is open-ended so that it may evolve into a resource management system. The system is an evolutionary one because: 1) consumers' (i.e., users of network resources) desires and requirements are unknown; 2) the impact of future, distributed operating systems on accounting is unknown; 3) radio and satellite adjuncts to the surface network will impact accounting; and 4) the interconnection of networks will involve accounting practices.

System Uses/Users

The accounting system involves three types of participants: ARPA, who provides and

controls budgets to accomplish specific research; facility managers, who provide computing services to support the contracted research; and customers, who use the resources. Accounting information for resource utilization may vary among installations, but in each case should be simple and minimal so as to aid rather than obstruct the administrative effort.

Resource utilization statistics and budget data should be collected and stored in a highly reliable and redundant system. This data should be transmitted over the network to the accounting facility and maintained there until required by the cost- and resource-reporting system.

System outputs are designed not only to report use, but to assist the customers, the facility managers, and ARPA in future planning by providing timely usage reports on a level of detail appropriate to the recipient. With regard to outputs, the system has built-in guards against subordinate compromise.

System Architecture

An accounting system is recommended that is partitioned physically and logically into two kinds of subsystems (see Figure 5.1). The first subsystem, the data collector, is duplicated (one on each Coast) for purposes of reliability. The data collectors are responsible for the fundamental, invariant properties of accounting. The second subsystem (of which there is only one), the report generator, evolves according to the changing needs of ARPA resource management.

The data collector receives as input new budget information from ARPA, suballocation of budget information from investigators, and resource usage reports from facilities. This

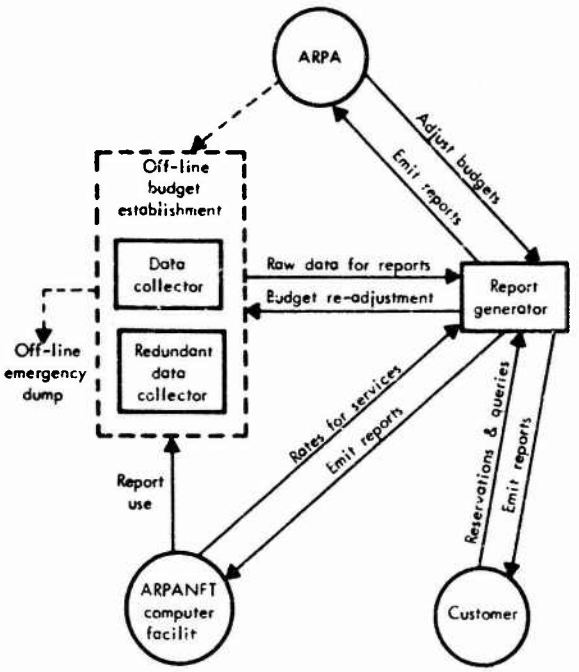

Figure 5.1 Accounting system configuration.

data is passed on demand to the report generator, with the latter periodically outputing reports; the generator can also be queried interactively over the network to produce reports, make reservations for resource use, and supply information on rates and services.

Implementation

The report discusses reliability and security considerations as well as proposing implementation milestones. A short design period was recommended in order to specify a model system, along with costs and an implementation schedule. Iteration and revision should produce a final system specification. A procurement and implementation phase would then result in a testable system. This would be followed by a checkout phase on the operational, experimental system.

Further Research

The study was reported . Dr. Lawrence G. Roberts, ARPA IPT, who discussed the need for network accounting at the December 1972 ARPA Contractors' meeting. A committee was formed to investigate and recommend accounting policies.

In addition to this effort, the Range Measurements Laboratory (RML), Patrick Air Force Base, has made arrangements with ARPA to assume some of the ARPANET management functions. The RCA Services Company, who provides technical assistance to RML, has visited ISI to discuss the study as it applies to their interests in ARPANET management functions.

DATA RECONFIGURATION SERVICE

An important goal of ARPANET work is to examine the fundamental intercommunications problems that arise in resource sharing among dissimilar systems. The Data Reconfiguration Service (DRS) [*18,22*] is a network experiment directed toward such an examination. The experiment involves communication between two arbitrary but cooperating processes with different input/output interfaces. That is, the DRS acts as a mediator, reformatting data passing between two superficially incompatible processes, to allow them to converse in their own data formats.

Other Approaches

Three approaches to solving such disparate communications requirements are:

1) service centers (servers) can tailor their software interfaces for coupling to a much larger set of users;

2) each user can provide the necessary software interfaces to all services he wishes to access;

3) high-level data-representation protocols, to which both users and servers conform, can be defined.

The first approach is highly unattractive because of the burdens and responsibilities it places on service centers. The second is likewise undesirable because it implies upgrading user equipment and modifying user programs to meet service center specifications. The inclination to date has been toward the third approach. Thus far, standards have been specified for logical message-path management, teletype-like character transmission, and file transfer.

DRS Approach

An interim (and perhaps even long term) solution to this communications dichotomy is the use of a fourth approach: the DRS. The DRS is a computer program, transparent to both user and server, that couples user and server and carries out transformations on data passing between them. (See Figure 5.2.)

This approach offers several advantages. Because the reconfiguration definitions (called

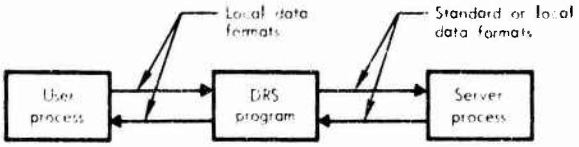

Figure 5.2 Data reconfiguration service schematic.

forms) are easily specified, user/server interface connections can be readily accomplished, with only minor changes made to their respective programs. For n x m transformations (n users times m services), there need be only a single adaptable transformer in the ARPANET.

An Operational DRS

Six ARPANET sites participated in the definition of a DRS language. From these specifications, ISI and The Rand Corporation developed a DRS compiler [22] and the University of California at Santa Barbara developed an interpreter and a network runtime package which combined to offer a real time DRS service on UCSB's IBM 360/75. The compiler takes character string definitions of data transformations and produces an intermediate (compiled) representation of the definition. The interpreter applies the compiled definitions to data streams passing between user and server in real time.

The service is now operational at UCSB on an experimental basis and has been used by ISI and UCSB. Other planned uses will be reported via the Stanford Research Institute, Network Information Center, Request for Comments mechanism.

Research Trends

Results of the DRS experiment are inconclusive at this moment because the service has been operational only a short time. However, several observations can be made. The overhead in CPU costs for the software version of DRS would probably prohibit its use for large volume data-handling applications. As a computer program, the DRS can be most effective in operating a one-time-only data reformatting service, where the original data is in one or more formats and where writing programs to reformat the data would be time consuming. From the standpoint of economics, a version of the DRS could be implemented in hardware if sufficient use at UCSB warrants.

PORTABLE TERMINALS

Development of a portable terminal for use with the ARPANET is in progress. The terminal will provide travelers with a device able to communicate from any telephone to any ARPANET site. That is, mobile users can send and receive mail and messages, as well as using other ARPANET services.

Size and Weight.

The terminal will be housed in a small briefcase (approximately 10" x 14" x 6") for portablility, and will weigh about 20 pounds. In contrast, conventional, complete portable display terminals are bulky, hard to carry, and weigh about 35 pounds. Off-the-shelf components are being used in the packaging effort to facilitate prototype development and evaluation.

Standard Handset.

The terminal will connect to any standard telephone handset and, via the telephone line, to any 300 baud Bell 102 compatible full-duplex, acoustic coupler dial-up unit. There will be an RS-232 EIA interface for fast, local, direct access.

Keyboard and Display.

The terminal uses an electronic Hall-effect 53 key keyboard with a Model 33 character set and a Burroughs self-scan panel display unit with 8 lines of 32 characters (256 character display). The display is alphanumeric upper case only. The terminal transmits and receives the entire USASCII character set.

Second Generation Portables

The completion date for the prototype is July 1, 1973. In a longer range effort, several alternative display devices are being evaluated. The ultimate goal is to produce a 2,000 character display terminal having the same capabilities and portability of the model described above. An 80 character per line, 25-30 line display is desired with an active viewing area of about 40 square inches (8" x 5"). The Owens-Illinois plasma panel and the conventional CRT can feasibly satisfy this requirement; no other devices have the required resolution (dot spacing of about .015"). Because of the advantages of gas panels (relative shock and vibration immunity, higher contrast (80:1), etc.) over CRTs, the gas panel displays will be carefully evaluated for the second generation portable terminal.

The completion of the second generation terminal depends on the availability of a suitable display device; delivery could be as early as several months or as long as a year. Efforts will continue in encouraging development of this type of display to fill the need for such a portable terminal. Means are also being sought to significantly reduce the weight and size of the unit. The current effort with the 256 character terminal will produce an evaluation instrument that will aid in defining the minimum terminal.

REFERENCES

1 McQuillan, J., W. R. Crowther, B. P. Cosell, D. C. Walden, and F. E. Heart, "Improvement in the Design and Performance of the ARPA Network", *AFIPS Conference Proceedings*, 1972 Fall Joint Computer Conference, Vol. 41, AFIPS Press, Montvale, New Jersey, 1972, pp. 741-754.

2 Atal, B. S., and M. R. Schroeder, "Adaptive Predictive Coding of Speech Signals", *Bell System Technical Journal*, Vol. 49, No. 8, October 1970, pp. 1973-1986.

3 Atal, B. S., and S. L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave", *Journal of Acoustical Society of America*, Vol. 50, No. 2, 1971, pp. 637, 655.

4 Maksym, J. N., "Real Time Pitch Period Extraction by Adaptive Prediction of the Speech Waveform", *1972 Conference on Speech Communication and Processing: Conference Record*, IEEE, New York, 1972, pp. 70-73.

5 Oetting, J. D., "A Formant Extraction Technique Using Predictive Coding", *1972 Conference on Speech Communication and Processing: Conference Record*, IEEE, New York, 1972, pp. 74-76.

6 Makhoul, J., "Aspects of Linear Prediction in the Spectral Analysis of Speech", *1972 Conference on Speech Communication and Processing: Conference Record*, IEEE, New York, 1972, pp. 77-80.

7 Markel, J. D., "Automatic Format and Fundamental Frequency Extraction From a Digital Inverse Filter Formulation", *1972 Conference on Speech Communication and Processing: Conference Record*, IEEE, New York, 1972, pp. 81-84.

8 Zebo, T. J., and W. C. Lin, "On the Accuracy of Formant Parameter Estimation Based on the Method of Prony", *1972 Conference on Speech Communication and Processing: Conference Record*, IEEE, New York, 1972, pp. 85-88.

9 Nakatsu, Y., A. Ichikawa, and K. Nakata, "Evaluation of Various Parameters in Spoken Digits Recognition", *1972 Conference on Speech Communication and Processing: Conference Record*, IEEE, New York, 1972, pp.101-104.

10 Mermelstein, P., "Speech Synthesis with the Aid of a Recursive Filter Approximating the Transfer Function of the Nasalized Vocal Tract", *1972 Conference on Speech*

Communication and Processing: Conference Record, IEEE, New York, 1972, pp. 152-155.

11 Kohda, M., and S. Saito, "Speech Recognition by Incomplete Learning Samples", *1972 Conference on Speech Communication and Processing: Conference Record*, IEEE, New York, 1972, pp. 311-314.

12 Itakura, F., and S. Saito, "On the Optimum Quantization of Feature Parameters in the PARCOR Speech Synthesizer", *1972 Conference on Speech Communication and Processing: Conference Record*, IEEE, New York, 1972, pp. 434-437.

13 Makhoul, J. I., and J. J. Wolf, "Linear Prediction and the Spectral Analysis of Speech", *Bolt Beranek and Newman Report No. 2304*, Cambridge, Massachusetts, August 1972.

14 Wakita, H., "Estimation of the Vocal Tract Shape by Optimal Inverse Filtering and Acoustic/Articulatory Conversion Methods", *Speech Communications Research Laboratory, Monograph No. 9*, Santa Barbara, California, July 1972.

15 Markel, J. D., "Digital Inverse Filtering--A New Tool for Formant Trajectory Estimation", *IEEE Transactions on Audio and Electroacoustics*, Vol. AU-20, No. 2, June 1972, pp. 129-137.

16 Haskew, J. R., J. M. Kelly, R. M. Kelly, and T. H. McKinney, "Results of a Study of the Linear Predictive Vocoder", *Record of the Southwestern IEEE Conference*, April 1972, pp. 340-344.

17 Markel, J. D., and A. H. Gray, Jr., "On Autocorrelation Equations as Applied to Speech Analysis", *IEEE Transactions on Audio and Electroacoustics*, Vol. AU-21, No. 2, April 1973, pp. 69-79.

18 Cerf, V., E. F. Harslem, J. F. Heafner, R. Metcalfe, and J. White, "An Experimental Service For Adaptable Data Reconfiguration", *IEEE Transactions on Communications Technology, Special Issue on Communications*, June 1972.

19 Balzer, R. M., *Ports: A Method for Dynamic Interprogram Communication and Job Control*, The Rand Corporation, R-605-ARPA, August 1971.

20 Teitelman, W., D. G. Bobrow, A. K. Hartley, and D. L. Murphy, *BBN-LISP TENEX Reference Manual*, Bolt Beranek and Newman, Inc., Cambridge, Massachusetts, August 1972.

21 Ellis, T. O., L. Gallenson, J. F. Heafner, and J. T. Melvin, *A Plan for Consolidation and Automation of Military Telecommunications on Oahu*, USC/Information Sciences Institute, RR-73-12, May 1973.

22 Harslem, E. F., J. F. Heafner, and T. D. Wisniewski, *Data Reconfiguration Service Compiler: Communications Among Heterogeneous Computer Centers Using Remote Resource Sharing*, The Rand Corporation, R-887-ARPA, April 1972.

PUBLICATIONS

1 Ellis, Thomas O., Louis Gallenson, John F. Heafner, and John T. Melvin, *A Plan for Consolidation and Automation of Military Telecommunications on Oahu*, USC/Information Sciences Institute, RR-73-12, May 1973.

2 Heafner, John F. aad Eric F. Harslem, "Large-Scale Sharing of Computer Resources", *1972 Wescon Technical Papers: Computer Networks-Session 7*, 1972.

3 Heafner, John F., "ARPANET Accounting System Study", January 1973, (informal report).

ACTIVITIES

1 Ellis, Thomas O., COTCO briefing (see Publications 1 above), to Brig. Gen. Howard McCormick and CINPAC Staff Hawaii, at ISI, April 16, 1973; to Commander Higgins, Department of the Navy, at ISI, April 18, 1973; and to Dr. Lawrence Roberts, et al., ARPA-IPT, in Washington, D. C., May 10, 1973.

2 Heafner, John F., and Eric F. Harslem, "Large-Scale Sharing of Computer Resources", at Western Electronic Show and Convention, Los Angeles, California, September 19-22, 1972.

(Cont.)

3 Richardson, Leroy C., member of working groups on Protocols, and Interprocess Communication, at Workshop on Automated Resource Sharing on the ARPANET, Cambridge, Massachusetts, May 21-22, 1973.

4 Richardson, Leroy C., member of User Interest Group (USING), Cambridge, Massachusetts, May 22-23, 1973.

6 PROGRAMMED AUTOMATION

Project Leader: **Robert H. Anderson**

Research Staff: **Ernest M. Hinds**
Gopal K. Kadekodi
Nake M. Kamrany
Bennet P. Lientz
Elliot A. Ponchick

Consultants: **Michael Boretsky**
Alexander F. Brewer
Vernon Edwards
Jack Rosenberg

Research Support Staff: **Sandra F. Whitaker**

INTRODUCTION

The goals of this project during the past twelve months have been to: 1) evaluate the technological feasibility of significant advancements in computer-based manufacturing systems for discrete product manufacture; 2) evaluate the economic impact on DOD and the U.S. economy derived from implementation of those advancements; and 3) define the development program required to achieve those advancements, including areas to be addressed and resources required. Those goals have been achieved. The evaluations and recommendations have been given to ARPA in a final report on the study phase of this project [1].

A major economic analysis has been performed of the cost, labor, and industry-structure factors characterizing DOD procurement; in addition, several detailed case analyses have been made of particular DOD-related manufacturing operations in order to understand the manufacturing enterprise as a system.

The Department of Defense procures over $20 billion worth of discrete goods each year; the great majority of these goods are sophisticated, precision products -- often electronic-based -- procured in rather small quantities. They are predominantly batch-produced. It is precisely in this category of discrete manufactured products that: 1) the U.S. has a comparative advantage in world markets; and 2) DOD requires healthy, efficient, national manufacturing support.

Batch manufacture of discrete precision products is characterized by a dynamic environment with continually changing priorities and job mix. Computers are not now being used effectively in this type of manufacturing environment to provide accurate up-to-date status information for management and for allocation, control, and monitoring of production resources. This is due to the inflexibility of existing software, the lack of interfaces natural to managers in this environment, and insufficient real time status reporting from the production processes.

A wide range of potential computer-based manufacturing systems have been examined which might increase manufacturing productivity, ranging from near-total programmable automation of manufacturing -- including

assembly and inspection processes -- to incremental enhancements to existing systems. The conclusion has been made that the most important development required by U.S. industry, and by DOD related industries in particular, is flexible, real time software systems that can be used directly by a manager and modified by him. As a consequence, these systems must have some limited modeling, deductive, decision-making, and specialized language-understanding capabilities. The development of a demonstration system having these features is recommended: components of such a system can be a distributed resource, shared among several users and accessed by them on a demand basis. Effective control software for complex batch production environments is a necessary precursor to efficient use of other advanced automation technologies, such as automated assembly and inspection systems. The primary direct benefits of an effective batch production management system are: 1) reduced inventory costs; 2) greater productivity through higher utilization of resources; and 3) better ability to meet schedules resulting in less slippage, defaults, and cost overruns. Due to the high degree of concentration of DOD procurement in a few major industries, advanced computer-based manufacturing systems introduced into these key industries can have a major impact in a relatively short time.

It is recommended that research also be continued in computer-controlled manipulation of objects, with particular emphasis on the mechanical dexterity required in assembly of avionic packages and other electronic-based units. Assembly and testing operations of this scale will not be eliminated by the continuing revolution in electronic components, and will be of increasing importance in batch-made products procured by DOD.

The batch production manufacturing environment requires a sophistication in computer-based resource control systems which is not available commercially, but elements of which have been demonstrated in research laboratories, particularly within the ARPA contractor community. These research capabilities must now be transferred to practical, commercial use.

IMPORTANT CHARACTERISTICS OF DOD PROCUREMENT

The characteristics of both the manufactured goods bought by DOD and the industries supplying those goods differ markedly from the characteristics of the goods and industries in the rest of the U.S. economy. The following issues summarize the results presented in reference [1]; these issues must be considered in formulating an R&D program which will have a major impact on the productivity of DOD suppliers.

Issue 1: The purchasing power of DOD's procurement budget is declining.

The budget of the Department of Defense can be considered as comprising two major components: discretionary funds (such as procurement*, over which DOD has yearly budgetary control); and nondiscretionary funds (such as personnel salaries, over which there

*The procurement appropriations of DOD finance the acquisition of capital equipment, such as aircraft, missiles, ships, combat vehicles, weapons, torpedoes, munitions, and communications; major items for support of the capital equipment when it is in use; the industrial facilities necessary to produce that equipment; and major modification of equipment in inventory where modernization can be achieved without buying new equipment.

is little immediate control). Figure 6.1 shows the breakdown of DOD's budget for FY 1972 into these categories and their major subcomponents.

During the past ten years, the portion of DOD's budget allocated to procurement has declined:

1) as a percentage of the DOD budget (down 10%);

2) as a percentage of the Federal budget (down 7.4%);

3) as a percentage of the GNP (down 1.3%).

If inflationary effects are considered, the real purchasing power of DOD's procurement budget during the last decade has declined 26 percent.

● Conclusion 1: It is vital that DOD obtain more value per procurement dollar.

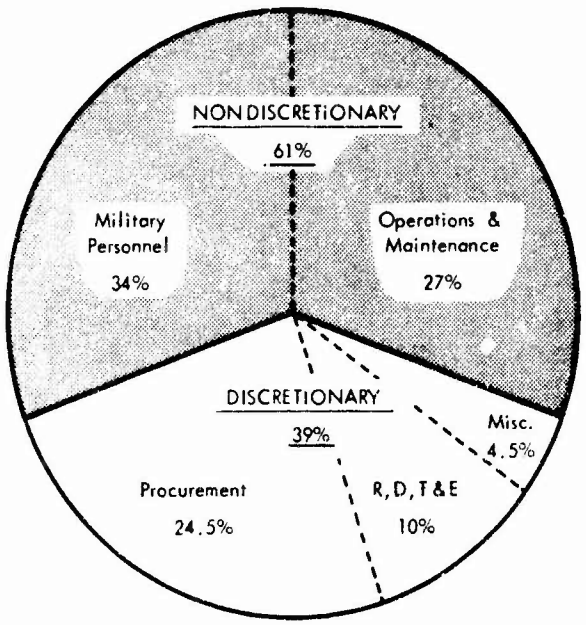

Figure 6.1 DOD budget FY1972. Total DOD budget, FY1972 = $76.7 billion. Source: The Budget of the U.S. Government, FY1972.

Issue 2: DOD procurement is characterized by relatively small quantities of sophisticated items, manufactured primarily by batch production methods.

Figure 6.2 shows a breakdown of DOD procurement by number of major end-items purchased. This chart was derived from the Procurement Annex of the Five-Year Defense Plan. On one hand, it understates actual production quantities of subcomponents, since there is some duplication of parts; on the other hand, the procurement budget does not take into account the significant DOD expenditures for research, development, test, and evaluation (R,D,T&E) in which large sums are spent on the construction, in very small quantities, of prototypes.

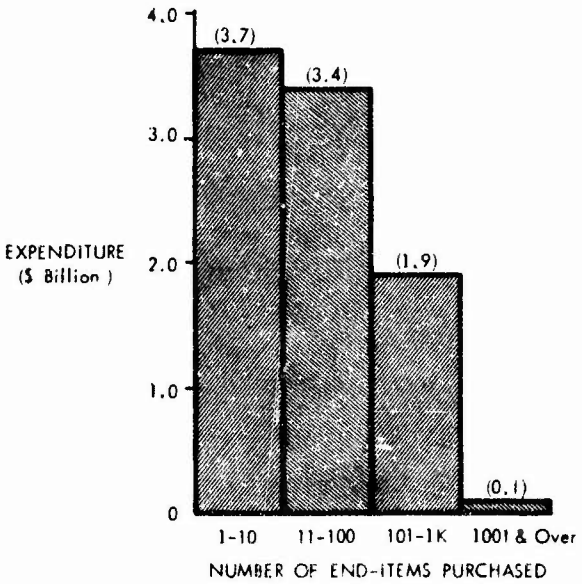

Figure 6.2 DOD procurement by number of major end-items purchased. Source: Procurement Annex, Five-Year Defense Program (FYDP).

The low production quantities shown by Figure 6.2 dictate that batch production methods be used by most major DOD suppliers and their subcontractors. (See [2] for a discussion of the distinction among batch production, mass production, and process control manufacturing methods.)

• Conclusion 2: Developmental programs aimed at increasing the productivity of DOD suppliers should concentrate on the batch production manufacturing environment.

It should be noted that the prime example of past DOD sponsorship of manufacturing technology, namely numerically-controlled machine tools [3] was aimed explicitly at batch production techniques.

Issue 3: DOD procurement is heavily concentrated in a few major industry categories.

Table 6.1 summarizes major DOD procurements by program. Aircraft and missiles alone account for over 50 percent of DOD procurement; with ships, they account for over 75 percent of procurement.

• Conclusion 3: Proposed defense productivity enhancement techniques should directly address problems in the aerospace industries and their suppliers, with a product mix characterized by: 1) precise, high performance, costly materials and assembly processes, and 2) electronic-based subsystems.

Issue 4: DOD procurement is heavily dominated by a few corporations.

Table 6.2 shows the top ten contractors for FY 1972. The prime contracts which they control account for 35.1 percent of all DOD procurement. The top 100 contractors have prime contracts accounting for 72.1 percent of DOD procurement, and the amount of contract award concentration in the top 100 contractors is increasing.

• Conclusion 4: 1) It is possible for DOD to have a substantial amount of contractual leverage in promoting the adoption of new manufacturing techniques; and 2) any proposed productivity-enhancement program should directly address the needs of these important suppliers and their subcontractors.

Again, the history of the transfer of numerical control technology [3] into industry emphasizes the unique role DOD can play in sponsoring new manufacturing technologies, due to the concentrated control it has over its suppliers.

Issue 5: The industry cost and labor structure for major DOD suppliers is significantly different from other industry categories.

Table 6.3 compares the labor breakdown in the aircraft and missile industry, and the electronic equipment industry (both important DOD suppliers), with automobile manufacturers. It shows that the ratio of administrative, clerical, and sales* employees per production worker is three times more for aircraft and missile manufacturers than for automobile manufacturers.

There are certainly many complex reasons for the variance in labor breakdown between the aerospace and automotive industries: the unique reporting and documentation requirements of government contractors; greater testing and inspection requirements in aerospace

*"Administrative, clerical, and sales" is a U.S. Department of Commerce term encompassing managers, accountants, lawyers, librarians, editors, purchasing agents, bookkeepers, estimators, office machine operators, secretaries, shipping and receiving clerks, stock clerks, and similar functions.

Table 6.1

SUMMARY OF DOD MAJOR PROCUREMENT PROGRAMS
1971-1973

Program	1971	%	1972e	%	1973e	%
Aircraft	6.3	35	6.6	35	5.9	31
Missiles	3.3	19	3.4	18	3.7	19
Ships	2.3	13	3.0	16	3.6	19
Other	5.9	33	5.9	31	6.1	32
Total	17.8	100	18.9	100	19.3	100

e = estimated (in $ billion)

Source: The budget of the United States, 1973 Appendix, p. 296.

work; batch production versus mass production manufacturing methods. The point is not that one industry is better or worse than another, but that there are basic differences which should be taken into consideration.

• Conclusion 5: 1) The unique characteristics of DOD suppliers require a productivity-oriented development program tailored to their needs; and 2) two major high-cost areas which should be explicitly considered are:

a) administrative, clerical, and sales functions in the aircraft industries (one-quarter of all industry-related labor); and

b) assembly in the electronic component and subsystem manufacturing industries (42 percent of all industry-related labor).

Issue 6: DOD-related industries hold relatively large and expensive inventories.

The value of inventories for all manufacturing groups at the end of 1967 amounted to $84.4 billion. The major DOD-related industries (e.g., aircraft and parts, ships, and communications equipment) are among those with exceptionally high inventory costs, most of which are "work in process" and "materials"; the latter are in contrast to "finished goods" subject to external demand factors.

On the average, U.S. manufacturing industry holds about 15 percent of the value of shipments in inventory on hand; however, the following are representative ratios of inventory/shipments for DOD-related industries:

Communications Equipment	24%
Aircraft and Parts	35%
Shipbuilding	30%

The approximate cost to DOD per year for inventories on hand for DOD-related manufacturing is in excess of $1 billion.

- Conclusion 6: A reduction of in-process inventories in DOD-related industries can suhstantially reduce procurement costs.

CASE STUDIES OF KEY MANUFACTURING INDUSTRIES

During the past twelve months, members of this project have met with over forty representatives of industry, research groups, and the government, and have inspected over thirteen different manufacturing facilities throughout the U.S., covering a wide range of products and technologies. The following three facilities represent many of the varied job shop environments of importance in the manufacture of DOD products. Detailed case studies of these three manufacturing environments have been made, with the active cooperation of the line managers responsible for production. The facilities studied are:

1) TOW Missile Production, Tucson Division, Hughes Aircraft Company, Tucson, Arizona;

2) Manufacturing Department, Precision Products Division, Western Gear Corporation, Lynwood, California;

3) Numerical Control Fabrication Facility, Douglas Aircraft Company, Torrance, California.

In each study, a variety of possible computer-based system innovations was examined which might have a significant impact on productivity, from near-total programmable automation of the manufacturing process to incremental enhancements of existing machines or control processes. Each of these case studies is summarized below, together with conclusions about the potential impact of advanced, computer-hased manufacturing systems on the production of these products.

TOW Missile Production, Tucson Dirision, Hughes Aircraft Company

Due to the widespread use of electronics packages in products procured hy DOD, it was felt important to study the manufacture of a product having an electronic module. After examining several possihle candidates, the Army's TOW (Tube-launched, Optically-iracked, Wire-guided missile), manufactured by the Tucson Division of Hughes Aircraft Company, was picked as being representative of a large class of electronic-based military products. Figure 6.3 shows the various components of the TOW missile.

Some relevant data concerning TOW procurement:

1) Procured materials and subassemblies account for about 70 percent of the TOW factory cost; therefore, automation technologies introduced only at the Tucson plant could affect at most 30 percent of the cost.*

2) Engineering changes are a significant item. During the production engineering phase (1966-1968), about 34 engineering

*One might expect significant changes in make versus buy decisions if a high degree of computer based automation were introduced. However, many decisions to buy from suppliers are made on the basis of the design competence of those suppliers (for such components as gyros, batteries, actuation systems)

Table 6.2

THE TOP TEN DOD CONTRACTORS FOR FY 1972
(IN DOLLAR VOLUME OF PRIME CONTRACTS)

	Company	DOD Contracts*	% of Total
1	Lockheed	1.7	5.1
2)	McDonnell Douglas	1.7	5.1
3)	General Dynamics	1.3	3.9
4)	General Electric	1.2	3.7
5)	Boeing	1.2	3.5
6)	American Telephone	1.1	3.4
7)	Grumman Corporation	1.1	3.4
8)	United Aircraft	0.9	2.9
9)	North American Rockwell	0.7	2.1
10)	Hughes Aircraft	0.7	2.0
	Total	11.6	35.1%

*(in $ billion). These figures are rounded.

Source: "100 Companies Receiving the Largest Dollar Volume of Prime Contract Awards", DOD (OASD) Directorate for Information Operations.

Table 6.3

COMPARATIVE LABOR BREAKDOWNS IN THE AUTOMOTIVE, AEROSPACE, AND ELECTRONIC INDUSTRIES (1970)

	% Motor Vehicles & Parts	% Aircraft & Parts	% Radio & TV Communications Equipment & Electronics
Non-production employees			
- R&D and other technical	7	19	18
- Admin., clerical, sales	12	25	21
Totals	19%	44%	39%
Production employees			
- Machining	11	16	3
- Assembly	30	9	42
- All other	40	31	16
Totals	81%	56%	61%
a) Total employment in listed industry	720,200	644,900	990,200
b) Total employment created in all industries*	1,793,300	1,102,800	1,683,300
Ratio of b) to a)	2.5	1.7	1.7

*Totol employment created in all industries is estimated using productivity changes and input-output data on interindustry transactions.

Source: Census of Manufacturers, U.S. Department of Commerce.

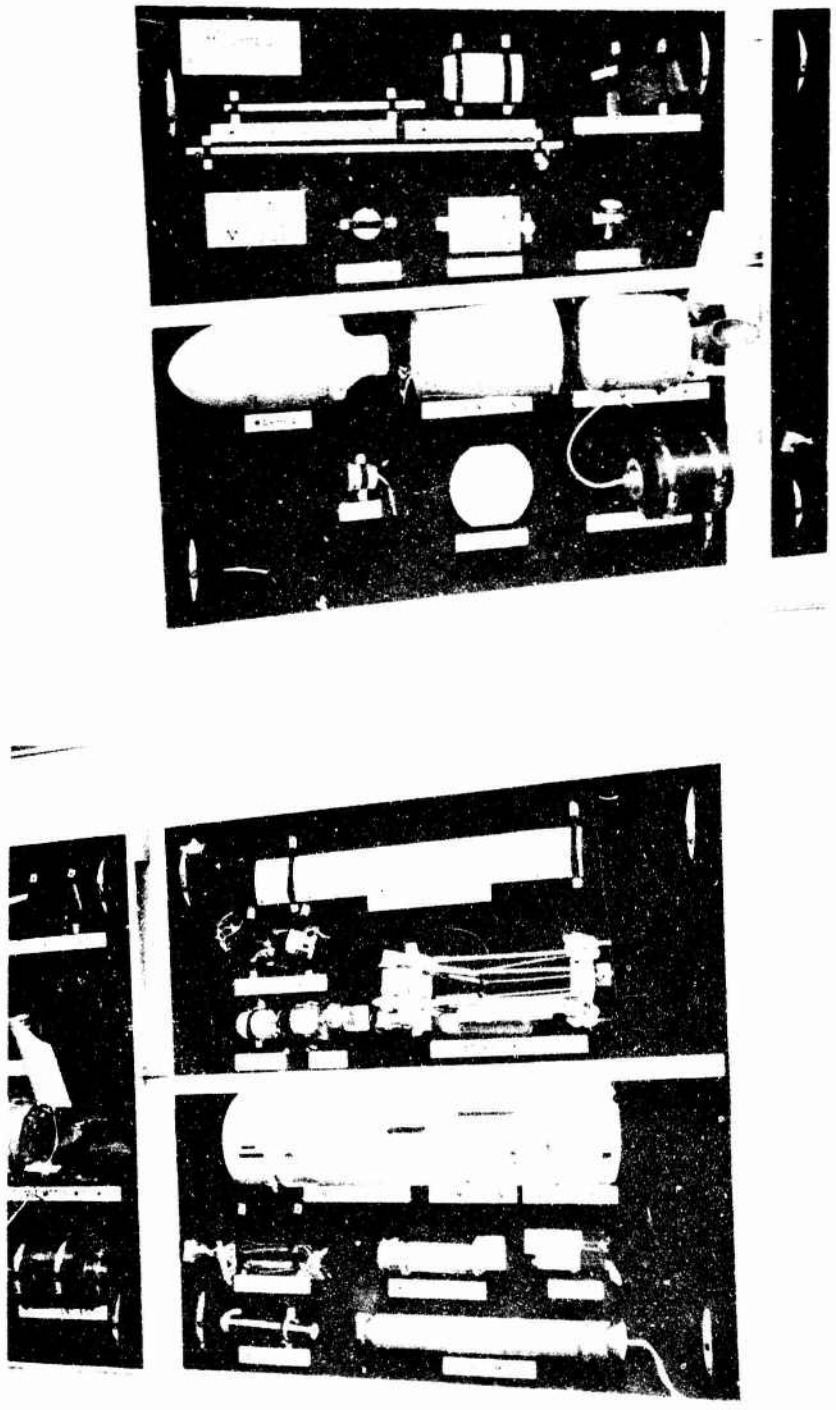

Figure 6.3 TOW missile components.

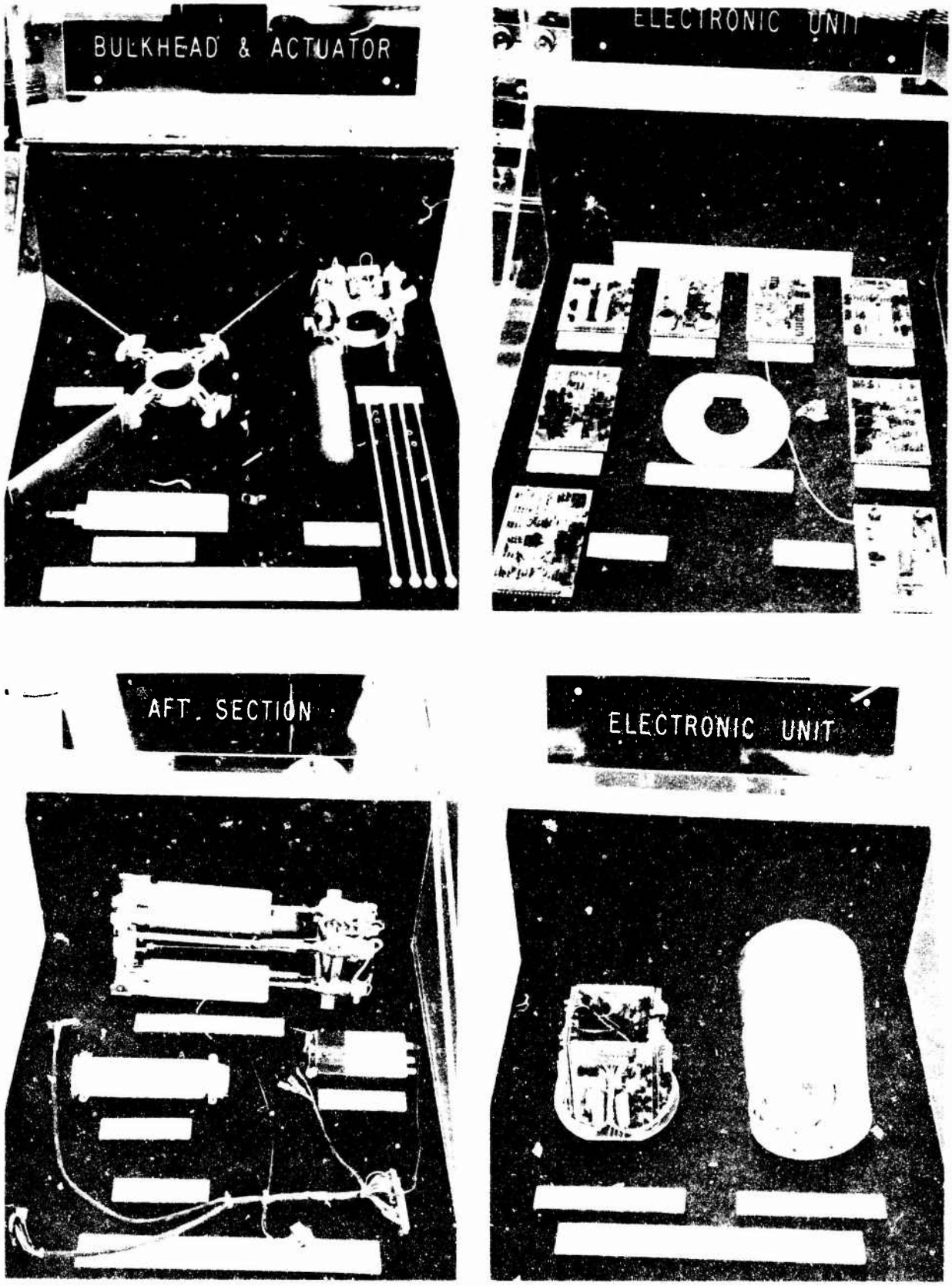

Figure 6.3, cont'd. TOW missile cross sections

change analyses were made each month. Each analysis defines a real problem and its engineering solution and cost. It may affect several different production drawings. By the end of 1971, after about eighteen months of normal production, there were still about fifteen analyses/month performed, and these resulted in about eight formal engineering change proposals per month, all requiring the approval of the Army Contracting Officer, because the change would affect the form, fit, function, or contract price of the product.

3) The overhead rate is about 130 percent. The cost to the government is about $12.50/hour (including G&A) per person on a program such as TOW. There are currently about 400 people at Tucson on the TOW program (including supervisors and other support personnel).

4) Rework costs can be significant. The most common cause is failure of electronic components, especially during vibration tests. The failure rate of the TOW electronic package during production is between 3 and 10 percent. (Only packages which eventually pass all such tests, of course, are assembled into the missile. After passing all such tests, the reliability of the TOW end product has been notable.) On a much more sophisticated missile such as Phoenix, about 30 percent of the labor is involved in rework during early production phases.

5) In October 1971, at a production rate of about 600/month, the unit price of a TOW missile was above $3,500. This was on a one-year contract. Currently, Hughes has a four-year guaranteed procurement contract. At the current production rate of 1500/month, the average unit price (including G&A and profit) is about $2,200. A major source of these savings has been significantly lower negotiated prices from suppliers, due to the four-year procurement potential.

Based on analysis of TOW and other electronic-based DOD procurement items, the following general observations have been made:

1) The continuing revolution in electronics (LSI, thin films, electron-beam etching, etc.) will not eliminate the assembly task in electronic packages. DOD's appetite for electronics seems to be constant volume, not constant function. When the space behind the warhead of a missile can hold an array computer, it probably will. The fact that a single chip will perform the functions of an entire circuit board will only lead to more complex designs involving many such chips. The space is there, and the increased functions can usually be justified.

2) The electronics revolution does not have a rapid effect on the production of military weapons systems. The design of the TOW, for example, was basically frozen in 1965 when the first prototype flew. Successful designs, having undergone elaborate testing and validation, are not usually changed in such fundamental ways as revamping the electronics based on next-generation possibilities. The hybrid circuit boards in the TOW will be around in it and in similar products for many more years.

3) Many of the subassembly tasks involved in the electronic and light source units seem potentially automatable, and they seem to absorb a disproportionate amount of assembly labor at present (about one-third of total assembly labor). The assembly operations required for such subassemblies are the most

promising yet seen for the potential application of programmable assembly machine techniques.

Manufacturing Department, Precision Products Division, Western Gear Corporation

Western Gear Corporation is the largest independent manufacturer on the West Coast of such precision mechanical products as gearboxes and actuators. Almost every DOD prime contractor manufacturing aircraft, ships, or missiles subcontracts to Western Gear.

The Precision Products Division at Torrance, California is an excellent example of a job shop: at any specific time, about 3,000 diverse Manufacturing Orders are in process. The case study focused on two aspects of manufacturing at Western Gear: 1) the detailed processes involved in their manufacture of one specific gearbox; and 2) the system by which a manager allocates and controls manufacturing resources and jobs in this complex job shop environment. The gearbox analyzed is shown in Figure 6.4; four of them are used by Lockheed in each C-130 aircraft.

Each box contains the following twelve major components, each of which is fabricated by Western Gear from materials purchased outside:

QUANTITY	PART DESCRIPTION
1	Housing - Main
1	Cover - Housing
1	Retainer - Bearing
1	Shaft - Input
2	Liner - Bearing
1	Sleeve - Bearing
1	Gear - Bevel
1	Adapter - Bearing
1	Gear - Output
1	Gear - Intermediate Spur
1	Spacer - Bearing

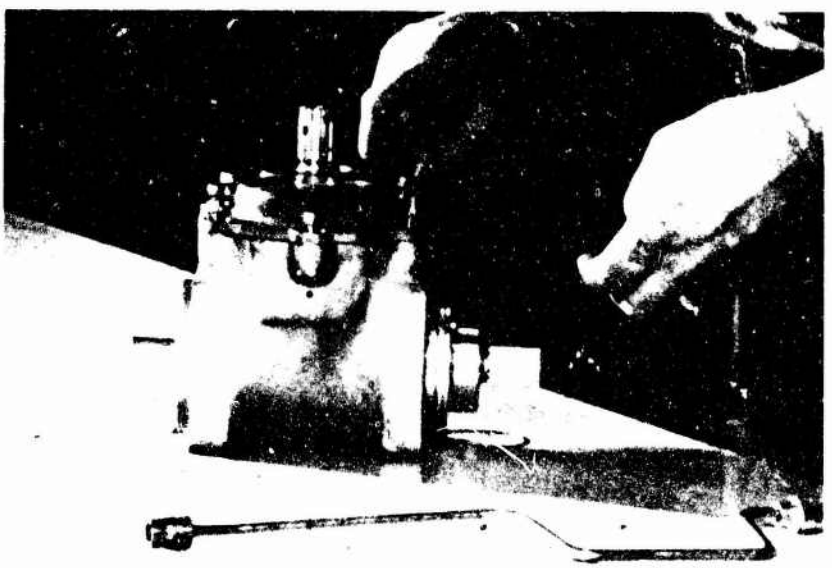

Figure 6.4 Gearbox manufactured by Western Gear Corporation for Lockheed C-130 aircraft.

For the fabrication of these twelve parts, and the assembly and test of the gearbox, about 26 man-hours of direct labor are required, about 75 percent of which are involved in direct operations and 25 percent in setup. The percentage breakdown of direct operation labor cost by department is shown in Table 6.4.

The percentage breakdown of setup labor cost by department is:

DEPARTMENT	% OF TOTAL DIRECT SETUP LABOR
Heat treat	51
Plating	20
Sawing	9
Turning(rough)	6
Total Other*	14
	100%

*Casting stores (3%); grinding (2%); stores (2%); N/C machining (1.9%); turning - line (1.7%); small gear cutting (1.6%); large bevel gear cutting (0.7%); inspection (0.6%); spindle drilling (0.3%); milling (0.2%).

Perhaps the most significant statistic concerning the production of this simple gearbox is that *258 movements of parts between different locations are required.* Clearly, in a job shop environment the control of in-process parts, tooling, and materials is a major consideration.

All of the departments involved in the above manufacturing steps were inspected and the operations were discussed with the Manufacturing Manager. The following are some of the major observations and the conclusions drawn:

1) The operation in which this gearbox was least representative of average products at Western Gear was final assembly. Most gearboxes contain five or ten times as many parts, which increases the demands on the assembler for dexterity, precision, and judgment. He is called on to use additional skills, not the same skills more often.

2) There are so many inadequately controlled variables in the physical properties and

Table 6.4

DIRECT OPERATIONS LABOR BREAKDOWN, C-130 GEARBOX

Department Involved	% of Total Direct Operation Labor	# of Distinct Operations
Inspection	19	50
Small gear cutting	15	10
Grinding	14	44
Turning (rough)	13	19
(De)burring	10	17
N/C machining	7	9
Large bevel gear cutting	5.7	5
Heat treating	5.5	35
Assembly	4.6	4
Turning (fine)	4	11
Spindle drilling	1	4
Sawing	1	8
Milling	0.2	1
Totals	100.0%	217

processing of metals that few gearboxes would meet the stringent DOD specifications without the continuous use of ingenuity by operators to beat the odds. When part fabrication operations have been automated (e.g., with numerically-controlled machine tools), the cost of the machines easily justifies the relatively small additional expense of a human operator to monitor the machine's performance; his reaction to malfunctions detected by subtle patterns of sight, sound, smell, and touch sensations can save tens of thousands of dollars and weeks of downtime.

● Conclusion: Significant reductions in direct labor are not possible (for this and similar precise mechanical products manufactured to existing DOD specifications). The only hope for cost reduction through automation of existing manual operations for these products would seem to be a radically new design philosophy allowing very reliable, accurate products (such as airplanes and missiles) to be made from components whose specifications have considerable latitude; such a development is highly unlikely.

3) An operator can usually meet or exceed the time standards for the set up and performance of each process if the material, tools, information, equipment, and conditions are on time and correct. It is not now possible for management to control all these factors and meet these criteria in at least half the cases; nearly all significant deviations from time standards can be traced to these causes.

● Conclusion: The management and control of resources in a job shop environment is currently a major source of inefficiency. The investigation of computer based production control systems to aid in the management of these resources in a job shop environment must have high priority.

4) During January 1973, at the time this study was performed, the Manufacturing Department had about 500 employees, and monthly billings were about $2,000,000. The overhead rate charged was 180 percent; of this, about 80 percent represented rent and other occupancy factors, fringe benefits, and equipment depreciation. The remaining 100 percent represented management and support services, i.e., people. *For each man performing a direct operation there was another person trying to keep him usefully employed.*

● Conclusion: This case study confirms that overhead services for DOD-related manufacturing industries are a major cost factor. (The same conclusion was derived from the comparative statistics for "administrative, clerical, and sales" labor. See IMPORTANT CHARACTERISTICS OF DOD PROCUREMENT above.) It is again concluded that computer-based management and engineering aids to increase the productivity and efficiency of indirect labor are of great potential importance.

An analysis was made of the existing production control system at Western Gear, a package called SCOPE (Scheduling by Computer and Overall Production Evaluation). It was developed by the corporate computer center for use by Production Control.

SCOPE, like many other production control systems observed, is a batch system useful only for rough job scheduling and monitoring. The major input is a Manufacturing Order and its scheduled completion date, both prepared manually by Production Control. Its major outputs are: 1) shop dispatch cards, one

for each operation scheduled; and 2) semi-weekly reports for Production Control, Accounting, the Manufacturing Manager, and several managers reporting to him.

Reporting of work accomplished is manual. Conflicts for resources and other problems which produce deviations from scheduled completion dates must be resolved manually. No useful exception reports are available.

Analysis of the use of the SCOPE system produced the following observations:

1) The Manufacturing Manager is charged $15,000 per month for the service; for that he receives a set of "tab runs" twice a week. The information is too voluminous to be used effectively.

2) The Manufacturing Manager has come to believe that it is difficult to make small changes to the existing system to improve its accuracy or functional capability.

3) The Manufacturing Department is required by corporate policy to purchase all programming and processing services from the Computer Center, and is not permitted to seek alternatives outside the company.

4) The job shop environment is extremely dynamic; priorities change on a daily -- sometimes hourly -- basis. Management decisions are continually required, often based on data which is not available in any formal system.

5) Control of resources on an hourly basis is currently performed manually. Each morning the Manufacturing Manager prepares a new list of critical jobs and questions. From 7:00 a.m. to 8:30 a.m. he meets with key members of his staff to request the *accurate current* status of the jobs: what has been completed, what is in process, what problems prevent the schedules from being met, and where every critical part is located. His subordinates are instructed to gather information by personally touring the shops, not just relying on the computer listings. As the reports are presented to him, the manager manually determines job priorities and gives orders for the rescheduling of personnel and equipment to overcome the critical problems. The manager spends the rest of each day following the progress of these items, trying to anticipate what future bottlenecks he created with that day's decisions, and preparing instructions for the night shifts to correct deviations from his morning plans that occurred during the day due to unpredictable variables.

- Conclusion: There is a requirement for real time industrial information and control systems which:

  1) are timely, accurate, and reliable;

  2) are flexible and modifiable;

  3) interface directly to managers having the responsibility for control; and

  4) aid managers in predicting the consequences of actions and decisions.

To meet these requirements, a level of sophistication is required in real time, interactive software systems which is not currently available in an industrial environment.

N/C Fabrication Department, Douglas Aircraft Company

The analysis of the Numerical Control (N/C) Fabrication Department at Douglas Aircraft Company concentrated on the management control center currently in use there to control the many complex jobs and resources in that job shop environment.

Figure 6.5 shows some of the wall displays in the management control center. This system effectively portrays a massive amount of information in summary form. Because the number of relationships between the data are effectively infinite, little attempt at dynamic preplanning can occur. As a result, crises occur regularly. The occurrence of crises necessitates a daily meeting of the manager with shop personnel during which additional information is interchanged and solution strategies are developed.

The manual data center has made it more efficient for the manager to make and implement some decisions, but it supports only a part of his responsibilities. He still spends the first two hours every day meeting with his staff and reviewing the status of critical jobs from a checklist. The rest of the day he and his staff physically track down the location and condition of parts and supplies, give orders to move them, and instruct operators to change the priority of their efforts.

Analysis of this facility has supported the conclusions reached in the study of production at Western Gear Corporation: real time software systems currently available as management aids for job shop environments are too costly, too inflexible, and do not support the important management function of predicting consequences of incremental changes to existing schedules. An entirely new level of software sophistication, at a reasonable cost, is required. Increased productivity is possible through much more efficient utilization of resources, if truly effective job shop production control systems are developed. Initial studies show the information environment in job shop manufacturing process control is sufficiently finite to permit development of the needed systems using the latest state of the art software techniques.

RECOMMENDED DEVELOPMENT PROGRAM

These analyses have documented the unique aspects of manufacturing for DOD procurement, the important leverage that DOD has in introducing new manufacturing technologies, and the real needs of manufacturers for new technologies shown by discussions with line management personnel and by analyses of their operations.

Based on these studies, it was concluded that an R&D program in computer-based manufacturing systems can have an important impact on manufacturing productivity. This R&D program should be focused in two areas:

1) development of a flexible manufacturing process control system, and its test and evaluation by in-plant use;

2) continued research in the programmable manipulation of objects, with experiments designed to better understand the trade offs involved in assembly, material transfer, machine loading, and other tasks involving mechanical dexterity.

Each recommendation is discussed briefly below.

Flexible, Manager Oriented Manufacturing Process Control System

Process control normally refers to the control of continuous processes, such as oil or steel production. Process control systems must be highly reliable, real time systems capable of performing control functions, as well as monitoring. Our primary recommendation is

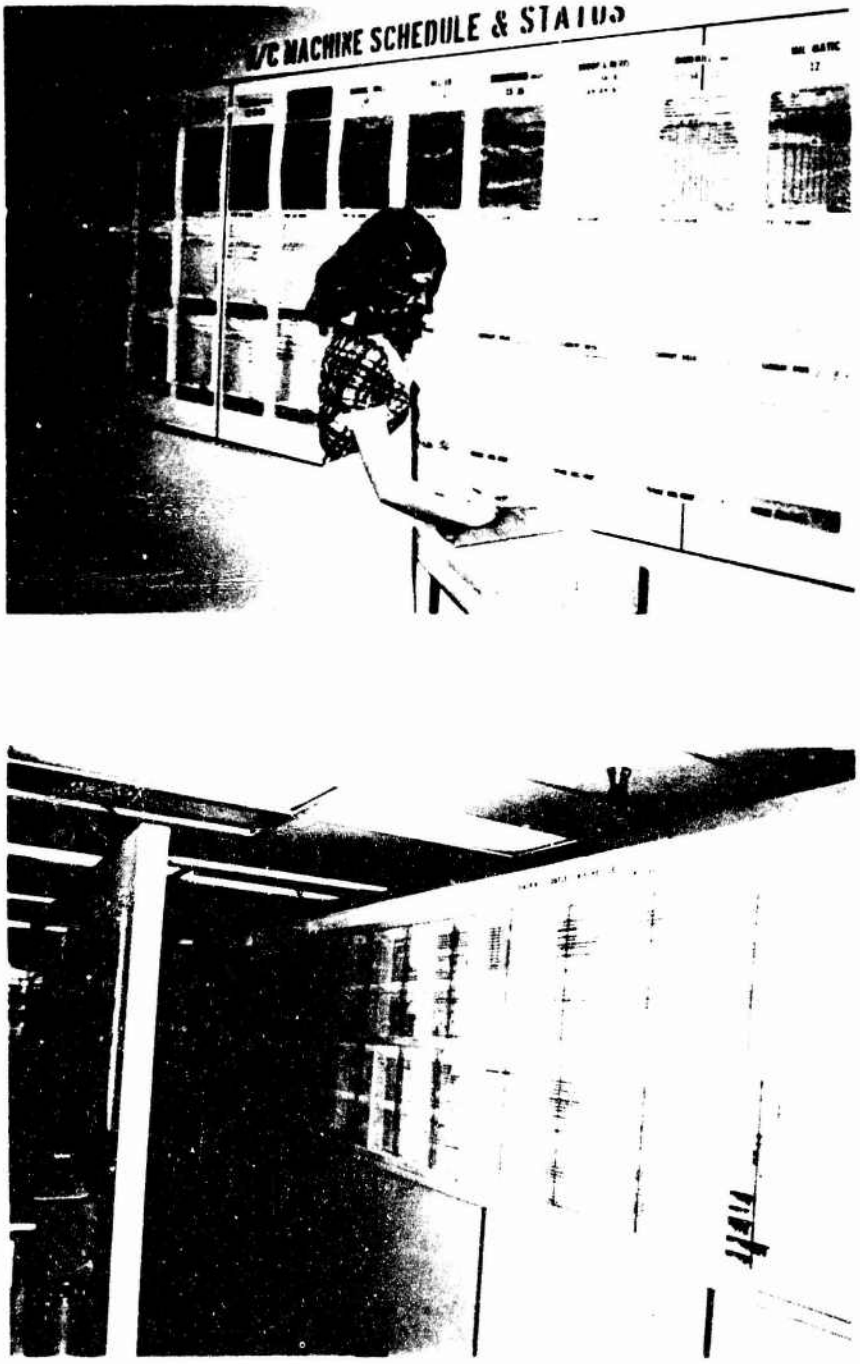

Figure 6.5 Wall Displays, Douglas Aircraft Management Control System.

that a system with these same attributes is vitally needed to assist in the control of the batch production of discrete manufactured products. The difference in the case of discrete manufacture (and the primary reason effective control systems do not currently exist in that environment) is that the complexity and dynamic nature of batch production require continuous, high-level decision-making by a human manager. What he needs to control his resources effectively is a computer-based control system with sufficient flexibility and an interface designed explicitly for his use; this system must also interface directly with his resources in that dynamic environment.

The important functional requirements of a Manufacturing Process Control (MPC) system are to:

1) give the manager accurate answers to common questions, such as the status of an order, the location of a part, etc.;

2) provide him or others exception reports when specified limit conditions have been reached, and enforce a user response;

3) provide status displays for critical operations, and enforce user response to certain conditions;

4) enable the manager to communicate with and control certain personnel, as well as leave a recorded audit trail;

5) allow the manager to control the allocation of key items of equipment directly with the computer;

6) allow him to explore the consequences of potential decisions;

7) be flexible, so that the manager himself can make small changes in its functons as his needs change without system redevelopment;

8) be usable directly by the manager, with an interface he finds natural to his task.

Where possible, the system should be capable of making subordinate decisions according to acceptable rules consequent to one of the manager's decisions. When adequate algorithms or heuristics exist for the system to make decisions independent of human intervention, the manager may decide that they should be incorporated into the system. This should require that other functions of the system be impacted minimally.

For the manager to maintain control of the plant, the system must automatically monitor the status of selected operations in the plant and report exceptions so that immediate action may be taken if necessary. Furthermore, for control to be maintained, the system should notify subordinates of actions to be taken at appropriate times and log the responses of the subordinates.

In order for the MPC system to have maximum transferability to a variety of industrial management environments, and to have sufficient reliability to be trusted by a manager as *the* process control system, it must adhere to the highest industrial software engineering standards:

- modular (independence of function, small blocks of code)

- program and data independence
- use standard languages
- independence from operation systems
- minimal use of machine-dependent code
- transaction-based
- interface to batch processing
- interface to other systems
- checkpoint/restart
- fail-soft
- self-monitoring
- completely documented
- thoroughly tested by people independent of the development group.

A Manufacturing Process Control system is proposed which is highly reliable and yet which incorporates flexibility and an interface tailored explicitly for use by a production manager. The key to such a system is the system architecture in Figure 6.6.

System A is stand-alone and performs real time data acquisiton, monitoring, and resource control functions. It must be highly reliable, fail-soft, and have the other attributes of a process control system. It has significant data acquisition, medium-speed storage, and user output capabilities, but a limited inquiry and command interpretation capability -- most probably with a highly stylized syntax. System A is capable of requesting assistance and receiving commands from System B. System B is potentially shared and potentially remote. It most probably relies on an internal conceptual model of the environment to perform a degree of specialized language interpretation and modeling.

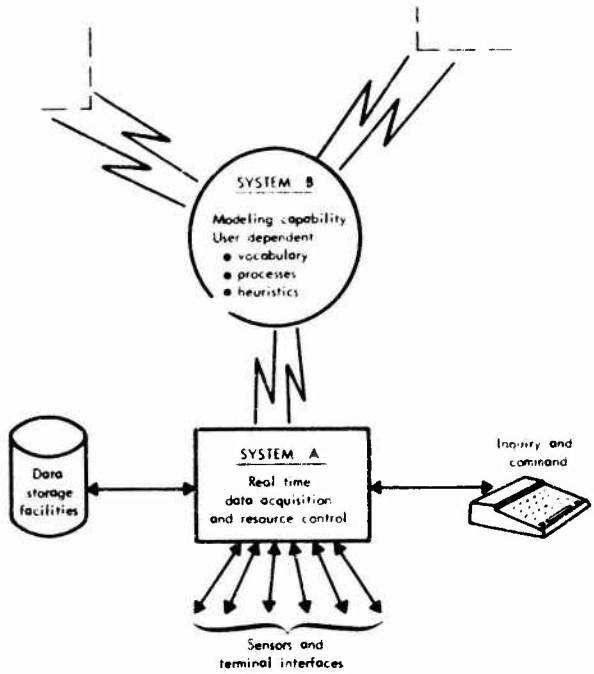

Figure 6.6 Proposed manufacturing process control system architecture.

When System A receives a query or command it cannot interpret, that input is passed on to System B; B is capable of requesting relevant data from A (possibly using the same limited query format as the terminal interface to A). The resources available in B are used on a demand basis. B's capabilities can be expanded as technology or demand permits, without compromising the integrity of System A. System B should utilize existing system components to the extent possible.

This framework permits the gradual introduction of user-oriented, flexible, interactive software into the manufacturing environment having the greatest potential payoff: a complex, batch-production manufacturing operation.

Research in Manipulation and Assembly

The second major recommendation is that ARPA continue a research program in computer control of machines applied to practical assembly and manipulative tasks in a factory environment. ARPA should be especially receptive to proposals for research on assembly tasks related to electronic-based units such as avionic equipment, small radar and communications units, and missile guidance units. Many reasons for this emphasis have emerged from this study:

1) Assembly workers constitute 42 percent of all employees in communications and electronic equipment industries.

2) DOD's demand for electronics appears to be constant volume rather than constant function; therefore significant assembly tasks will remain in spite of the continuing revolution in electronics technology.

3) The consistently small size and weight of electronic components simplify the mechanical requirements in constructing a prototype assembly machine, and the resulting machine(s) could be broadly applicable to many different electronic-based products.

4) A prime requirement in electronic fabrication is the integration of testing procedures; greater mechanization of the fabrication tasks would permit more timely, thorough, computer-controlled testing to be introduced into the fabrication process.

A flexible, programmable electronic unit assembly machine would be especially useful in the construction of prototypes; there is often a requirement for 10, or 50, or 100 identical units for testing (e.g., destructively, in a missile or smart bomb). It is currently very difficult to assure that such prototype units are, in fact, fabricated identically by hand. A preliminary examination of electronic equipment assembly operations indicates that this is also an area in which considerable computer-based machine flexibility is required. The nature of the physical processes involved leads to such conditions as accidental drops of solder causing unexpected circuit errors, and jiggling heuristics needed for proper plug and board insertions. (There is also an important need to explore redesign possibilities for plugs, sockets, and components to make them more amenable to manipulation by machine.) Much of the research on manipulation and machine dexterity performed in the area of electronic unit fabrication will be relevant to the more difficult and more varied task of mechanical assembly of discrete objects.

We strongly recommend that any R&D program in automation of assembly tasks be based on the validation procedures which we have used in the present study:

1) perform case analyses of electronics fabrication operations currently being performed, and obtain estimates, from manufacturing representatives with line responsibility, of the expected changes in fabrication technology to be utilized by their facility during the next decade;

2) form an Advisory Council of line manufacturing representatives to validate the conclusions of that study and the functional specifications derived from it for a prototype machine;

3) perform an R&D program in cooperation with representatives of relevant industry groups to develop prototype systems;

4) arrange for on-site test and evaluation of prototype systems in actual production environments.

FUTURE RESEARCH AND DEVELOPMENT

In the next year a development program at ISI has been proposed for a flexible management system for the control of production. It is intended that this system be broadly applicable to the control function in diverse batch production manufacturing environments, thus achieving the maximum possible effect on the productivity of major DOD suppliers. It is planned to draw upon the resources of the software system research being performed within the ARPA community of contractors, and to enter into explicit relationships with industries which will allow the transfer of the resulting technology for test and evaluation. It is expected that a System A will be operational in one or more factory environments within eighteen months; this initial "window" into the information and control flow in the factory will then allow the introduction, test, and evaluation of various System B advanced capabilities.

REFERENCES

1 Anderson, R. H., and N. M. Kamrany, *Advanced Computer-based Manufacturing Systems for Defense Needs*, USC/Information Sciences Institute, RR-73-10, July 1973.

2 Anderson, R. H., *Programmable Automation: The Future of Computers in Manufacturing*, USC/Information Sciences Institute, RR-73-2, March 1973.

3 Rosenberg, J., *A History of Numerical Control 1949-1972: The Technical Development, Transfer to Industry, and Assimilation*, USC/Information Sciences Institute, RR-73-3, (in progress).

PUBLICATIONS

1 Anderson, Robert H., *Programmable Automation: The Future of Computers in Manufacturing*, USC/Information Sciences Institute, RR-73-2, March 1973.

2 Anderson, Robert H., and Nake M. Kamrany, *Advanced Computer-based Manufacturing Systems for Defense Needs*, USC/Information Sciences Institute, RR-73-10, (in progress).

3 Rosenberg, Jack, *A History of Numerical Control 1949-1972: The Technical Development, Transfer to Industry, and Assimilation*, USC/Information Sciences Institute, RR-73-3, (in progress).

ACTIVITIES

1 Anderson, Robert H., "Computers and the Factory: Can Productivity be Dramatically Improved?", Computer Science Seminar, Stanford University, November 21, 1972.

2 Anderson, Robert H., "Machine Intelligence and Automation", University of California at Los Angeles, May 15, 1973.

7 TENEX FACILITY

Project Leader: **John T. Melvin**

Research Staff: **Thomas L. Boynton**
Stuart C. Feigin
Robert H. Parker
Leroy C. Richardson

Research Support Staff: **Judy Gustafson**
Mary J. Jaskula
Jerry Pipes

Student Aides: **Raymond L. Bates**
Danny L. Charlesworth
Ronald L. Currier
Jimmy T. Koda
Virginia E. Sato

INTRODUCTION

The ISI time sharing facility is operated as a research and service center in support of a broad set of ARPA projects. It currently services 400 users, 65% of whom access the facilities via the ARPANET from locations extending from the East Coast to Hawaii. All facilities of the operating system are available to all users, independent of whether they are locally or remotely connected through the ARPANET.

The system consists of a PDP-10 CPU, Bolt Beranek and Newman (BBN) paging box, large capacity memory, high-performance paging drum, on-line file storage, and associated peripherals (see Figure 7.1). The TENEX operating system, in conjunction with the large core and drum memory, provides considerable computing capacity and is capable of supporting a wide variety of simultaneous users.

To increase system availability, additions have been made in the hardware and software facilities and in support personnel. As a result, system usage, both by ISI and non-Institute users, has expanded greatly during the eight months the facility has been in operation.

Video Display System

A Video Display System is being developed to provide an inexpensive quality terminal for computer users (i.e., programmers, managers, secretaries) requiring significantly more capabilities than currently available with low cost terminals. These capabilities include: a full page of text (4,000 characters), graphics, 256 words of writable font, and a standard communications interface to facilitate computer connection. A modular design (part of the stated requirements) will allow components to be used in a clustered environment as well as in a remote stand-alone unit, with minimum cost differential. A Request for Proposal has been distributed to twelve vendors with responses of interest expected from at least four vendors by June 29, 1973.

FACILITY ESTABLISHMENT

The system was moved from another location in the beginning of October 1972. Several months prior to the move, considerable effort was spent preparing the Institute site for the

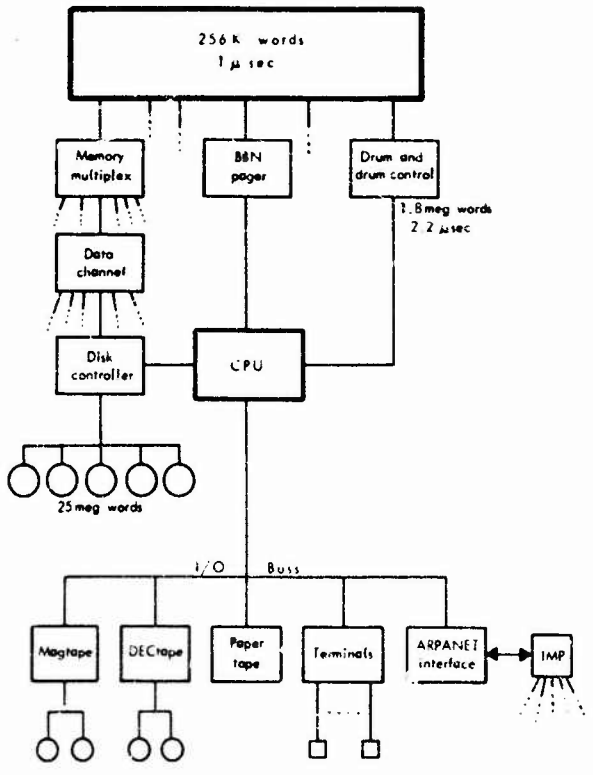

Figure 7.1 TENEX facility.

arrival of the computing equipment. Preparations included machine room layout, specification of all environmental and electrical requirements, overseeing of the installation of the specified support equipment, and the physical movement of the computing equipment. The site, including the computing system, became operational in early October.

GENERAL SYSTEM UPGRADING

Since moving to these facilities, considerable emphasis has been placed on increasing the availability of the system for both local and ARPANET users. To accomplish this task, additions have been made in hardware and software, and in support personnel.

Hardware

The hardware acquired since the move to ISI includes additional terminal access ports for local users, magnetic tape units for file backup and recovery, and an additional disk drive to expand the system's on-line storage capability.

Software

A joint effort was undertaken by ISI and the Network Information Center (NIC) at the Stanford Research Institute to modify the NIC archival facility for use on the TENEX here. The facility was implemented so as to allow the system operator to selectively archive, to magnetic tape, user files which have remained dormant within the system. A linkage was created within the file system so that the user can subsequently interrogate the archive and notify the operator of his desire to have the file retrieved. The current implementation is viewed as preliminary and incomplete. Additional capabilities and extensions are planned and under development. However, it is apparent that even the preliminary implementation is usable and does provide the needed extension to magnetic tape storage.

Another system software development effort concerned hardware diagnostics. Digital Equipment Corporation (DEC) peripheral diagnostics have been modified to run under the

operating system as privileged jobs, thereby reducing the amount of stand-alone time required to perform preventive and corrective maintenance. Other diagnostics have been modified or developed to aid in the identification and isolation of problems resulting from the interaction of DEC and ISI-built equipment.

Support Personnel

The Institute has hired and trained student operators to provide twenty-four hour machine coverage. The primary responsibility of the operator is to assist the system staff in maintaining the operating system and in the continuous functioning of the overall system. The on-duty operator is principally engaged in running system maintenance programs and monitoring machine room equipment. The latter function includes detecting failures on the computer or computer support equipment, taking corrective action, and notifying appropriate system personnel in the event of a nonrecoverable failure.

A full-time system programmer has also been hired, bringing the system programming staff up to one and one-half full-time personnel.

LOCAL PROJECT SUPPORT

The TENEX facility has been utilized extensively in support of local projects. The staff makes use of all of the available standard subsystems (e.g., editors, compilers, assemblers, and utilities). Additionally, staff members have written subsystems and utilities in support of their projects (e.g., a mailing-label program, photo typesetting support programs, TENEX accounting package extensions, and a special hard copy terminal controller). The facility has supported less frequently used subsystems at the special request of users.

XGP.

The installation of a Xerox Graphics Printer (XGP) is planned to provide for the formal printing and graphics requirements of the Institute. The XGP is essentially a video to Xerox hard copy device. The printer has been installed in several other locales, along with associated processors for converting coded information to video or raster data for the XGP. It is planned to increase the vertical and horizontal resolution over that currently in use at other sites in order to produce the high quality output desired. Since the inclusion of graphics will require more processing power than the PDP-11 support processors used at other sites, the XGP will eventually be driven with the speech processor described in the Network-Supporting Research section of this report. It is intended also that the XGP will be a natural part of the remote conferencing mechanism under development by the Network group.

Also, modifications to the operating system and/or special programs have been written in direct support of Institute projects.

MLP-900 Processor.

Monitor modifications to support the initial installation and check-out of the MLP-900 have been developed and verified. These modifications allow basic processor to processor communication through both the I/O buss and memory. This is step one of the multistep plan for fully integrating the MLP into the TENEX operating system. (See the Programming Research Instrument project report.)

Network Measurements.

The system has been utilized to obtain measurements of interest to the Network projects. (See the Network-Supporting Research project report.)

ARPANET USAGE AND SUPPORT

System usage by ARPANET users has increased significantly since the machine has been at ISI. Prior to moving to ISI, Network users numbered approximately 30. Since the move, the number has increased steadily. In January, 50 Network users utilized 30 directories; by March, 350 Network users utilized 230 directories. Beginning with May, approximately 400 users were using 300 directories.

All Network users are assumed to be experienced with the use of TENEX and the ARPANET. Institute operators and staff members have engaged in actively assisting Network users and will continue to do so. Of necessity, this assistance frequently must be confined to providing information on the location of documentation, or the names of other non-Institute individuals to contact when the question concerns non-Institute software.

VIDEO DISPLAY SYSTEM

A special project has begun to develop a Video Display System to replace the one formerly available with the TENEX time sharing system. This project will result in the procurement of a high quality terminal system for members of the Institute.

The terminals will be part of the user's office environment and will be used as any other tool (e.g., a telephone) by the staff in the performance of their duties. Most of the offices are in close proximity to a communications processor (which is interfaced to the ARPANET) in a clustered environment; however, a remote capability will be available.

The basic terminal consists of a high quality TV monitor, and a keyboard with required control logic. The terminal will be predominantly used for text (alphanumeric) communication with TENEX; however, a quality graphics capability is also required. The problem dictating the display system architecture is how to provide high quality graphics to a limited number of users without prohibitively taxing, in dollars or capabilities, the text-only users. The graphics requirement has two aspects: a limited and unlimited number of vectors. If economically feasible, all display terminals will be provided with a limited graphics capability. The graphics requirement assumes compatibility with a variety of input devices (e.g., light pen, tablet, mouse, etc.) which can be added to the terminal as required.

Display System.

The display system consists of digital generators (scan converter, character and vector generators) for a high resolution raster scan display. The raster scan is required to insure a flicker-free display without concern for the display content. The resolution of the system displays a full page of text (typed, single space, approximately 4,000 characters). The high resolution of the display is also needed for the quality of vectors desired. The stipulation of an all digital system is consistent with a requirement for maintainability and reliability. It is also assumed that a quantized

video display produces better images, simplifies the video distribution system and the TV monitor, takes advantage of current digital technology, makes more feasible an all solid-state approach, simplifies the registration problems with input devices, and is more compatible with a desired modular building block approach.

Minimal Centralization.

An additional display system design goal is production of a terminal that minimizes the required centralized equipment so terminal remoteness can better be achieved. The minimalized centralization reduces the per unit cost differential of single and multiple terminal clusters. For the most part, the required terminal configuration will be in close proximity to the interfacing communications processor. Economic considerations may dictate packaging some of the components in a central unit or multiplexing some of the logic. This economic advantage should not compromise the display terminal architecture in such a way that the stand-alone unit cannot be assembled from the same set of building blocks. The control commands, and character and vector order codes are identical for the local and remote terminals.

Building Blocks.

The clustered video terminal system is perceived as a collection of building blocks, all available on demand and shared by the users via automatic control. These blocks consist of 20 character and 20 vector generators, refresh memories, and interface controls (see Figure 7.2). The TV monitor has a raster format of 1,280 points per line and 864 visible lines. The monitor operates with either black or white video, with the predominant use to be black

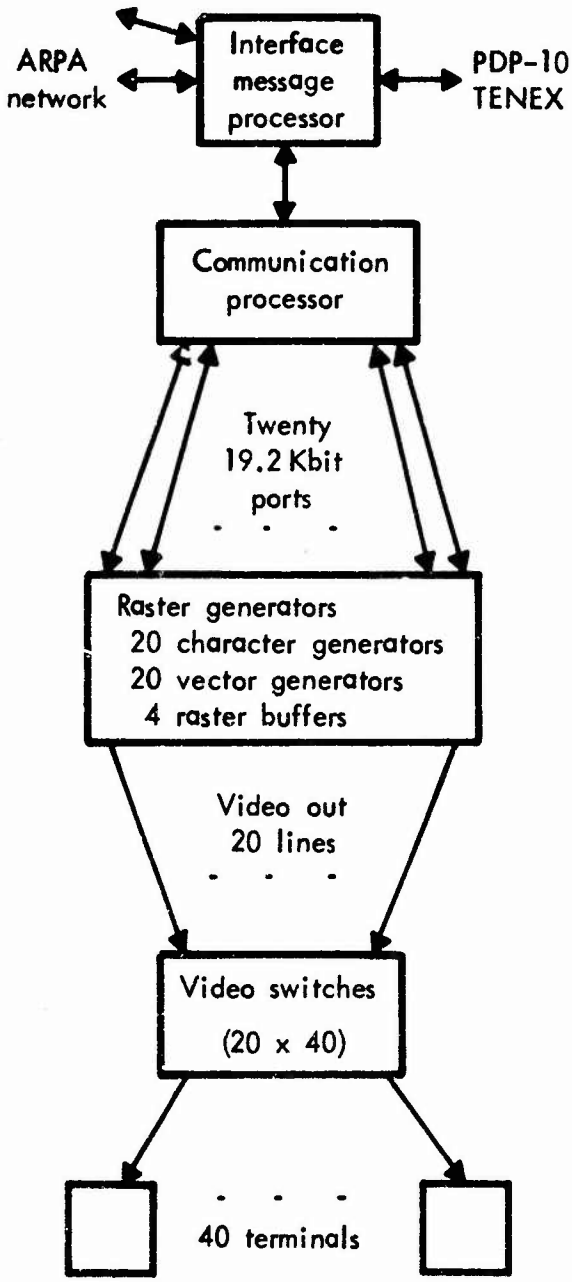

Figure 7.2 A clustered video display system.

on white. The generators are interfaced to the computer system with up to a 19.2 kilobits/second teletype-like (EIA RS-232C) channel. These ports will be provided by a communications processor interfaced to the ARPA Network. The generators are dynamically tied to the terminal through a video switch and a video distribution system as required by the user. The raster refresh memory is also an on demand resource and is only required for the full graphics user.

Survivability.

The display system resulting from this procurement will be a high quality state of the art instrument for the computer's user environment. In selecting a vendor to produce the display system, consideration will be given not only to his ability to produce this prototype system, but also to insure product survivability should the displays be in demand by others.

Expandable System

The performance of the terminals can be expanded in a variety of ways, depending on economic considerations. There are, however, additional highly desirable attributes for the Institute's applications which will be considered during terminal system development. These are:

- an open-ended system to allow for additional terminals or generators, without making existing hardware obsolete;
- a two-page display of approximately 8,000 characters on a single screen;
- variable character spacing up to a maximum of 100 characters per line;
- expansion of character capacity up to twice the present size;
- line width control, rather than gray level or color, for video modifiers.

The responses to the Request for Proposal will be reviewed during the month of July 1973. It is anticipated that a contract will be awarded during August 1973 with a one-year completion date. Partial deliveries of terminals are expected throughout the year to replace the Institute's currrently-leased terminals.

8

COLLOQUIA

May 1972 to May 1973

June

Harry Keirns, Gould Inc. Gould printer, June 29, 1972.

July

Robert Mazza, IBM. Data communication study, July 21, 1972.

Robert Anderson, ISI. Automation, July 24, 1972.

Donald Oestreicher, ISI. MLP-900 programming languages, July 27, 1972.

August

Martin Yonke, ISI. MLP-900, August 7, 1972.

David Parnas, Carnegie-Mellon University. Response to errors in well-structured programs, August 11, 1972.

Theodore Kehl, Washington University. On-line computer systems, August 21, 1972.

Gerald Sussman, MIT. Artificial intelligence, August 22, 1972.

Karl Balke, Burroughs Corp. Information systems performance theory, August 29, 1972.

September

Douglas Hogan, National Security Agency. MLP-900 speech recognition and security, September 12, 1972.

Jerry Elkind, Xerox Corp. Research programs at Xerox, September 13, 1972.

Theodore Glaser, Case Western Reserve University. Vocoders and video bandwidth reduction techniques, September 15, 1972.

Robert Merrell and Charles Seitz, Burroughs Corp. Video, September 20, 1972.

Ralph London, ISI. Program correctness September 21, 1972.

Vint Cerf, Stanford University, Network characteristics, September 29, 1972.

Wayne Wilner, Burroughs Corp. Burroughs 1700 microprogramming computer, September 29, 1972.

October

Richard Bisbey II, ISI. SAAB FCPU microprogrammed computer, October 18, 1972.

Warren Teitelman, Xerox Corp. BBN-LISP and run-time enhancements to ease programming, October 20, 1972.

Glen Culler, Culler-Harrison Inc. SPS-32 signal processing computer, October 26, 1972.

Robert Parker, ISI. Vocoder techniques, October 30, 1972.

November

Alan Kay, Xerox Corp. Teaching programming to children, November 3, 1972.

Walter Martin and Martin Schrage, Computer Signal Processors, Inc. CSP-30 floating point processor, November 3, 1972.

Albert Zobrist, USC. Advice-taking chess machine, November 6, 1972.

Donald Good, University of Texas. The NUCLEUS verification system, November 17, 1972.

Jerry McClellan, STANDARD Computer Corp. IC-4000, November 22, 1972.

December

William Pratt, USC. Image processing, December 1, 1972.

James Mitchell, Xerox-Palo Alto Research Center. Data extensions, December 4, 1972.

Earl Swartzlander, USC. The inner product computer, December 19, 1972.

January 1973

Don Blankenship, University of California at San Diego. Judgements of temporal sequence, January 4, 1973.

Judy Townley, Harvard University. ECL fundamentals, January 10, 1973.

Thomas Cheatham and James Forgie, Harvard University. Advanced concepts in utilization, January 11, 1973.

Jaimie Carbonell, Bolt Beranek and Newman Inc. Mixed initiative manned machine systems, January 12, 1973.

Rob Kling, University of Wisconsin. Fuzzy set theory with PLANNER, January 12, 1973.

Rob Kling, University of Wisconsin. Toward a person-centered computer science, January 12, 1973.

Karl Balke, ISI. Performance theory, January 15, 1973.

Ronald Resch, University of Utah. Computer sculpture/architecture models, January 25, 1973.

February

James Levin, University of California at San Diego. Models of human inference, February 5, 1973.

Patrick Baudelaire, University of Utah. Color perception, Fehruary 6, 1973.

Eduardo Fernandez, University of California at Los Angeles. Activity transformations in a graph model of parallel computations, February 7, 1973.

Barry Wessler, University of Utah. Computer animation, Fehruary 15, 1973.

Guner Robinson, ISI. Survey of speech processing techniques for vocoding purposes, Fehruary 20, 1973.

Guner Robinson, ISI. Details of the linear predictive coding techniques for speech compression, Fehruary 28, 1973.

March

Louis Mailloux, Xerox Corp. Presentation on the Xerox Graphics Printer, March 1, 1973.

James Morris, Jr., University of California at Berkeley. Protection in programming languages, March 7, 1973.

William Quirk, Naval Air Development Center. Mid-1960s Boeing Aircraft experiment on distributed project management-conferencing and its relation to ARPANET conferencing, March 7, 1973.

William Joyner, Harvard University. Automatic theory proving and the decision problem, March 8, 1973.

Anita Jones, Carnegie-Mellon University. Protection in programming systems, March 14, 1973.

Peter Pettler, DiPhase Corp. Presentation on portable terminals, March 15, 1973.

Larry Snyder, Carnegie-Mellon University. Analysis of programming languages using schemata, March 15, 1973.

Daniel Swinehart, Stanford University. A multiprocess approach to interactive programming systems, March 16, 1973.

Kenneth Modesitt, Purdue-Ft. Wayne. ELIJAH - an intelligent assistant for natural language programming, March 19, 1973.

Stephen Jones, Southern Methodist University. Adaptive filters, March 20, 1973.

Stephen Jones, Southern Methodist University. Speech processing, March 20, 1973.

James Watt, Burroughs Corp. Presentation on Self-Scan, March 22, 1973.

Thomas Moran, Carnegie-Mellon University. The symbolic nature of visual imagery, March 30, 1973.

April

Robert Furey, Lockheed Corp. Presentation on SUE computers April 11, 1973.

William Howden, University of California at Irvine. Naive problem solving, April 18, 1973.

May

Alexander Christakis, Academy for Policy Analysis. Socio-planetarium, May 7, 1973.

Norton Greenfeld, California Institute of Technology. Optimization in relational data base systems. May 8, 1973.

Leon Stucki, McDonnell-Douglas Astronautics Co. Automatic generation of self-metric software, May 9, 1973.

David Wile, Carnegie-Mellon University. A generative, nested-sequential basis for general-purpose programming languages, May 10, 1973.

Walter Ryder, USC. The role of formal decision-making methods in interactive management systems, May 11, 1973

9 PUBLICATIONS

1 Anderson, Robert H., *Programmable Automation: The Future of Computers in Manufacturing*, USC/Information Sciences Institute, RR-73-2, March 1973; also appeared in Datamation, Vol. 18, No. 12, December 1972, pp. 46-52.

2 ---, and Nake M. Kamrany, *Advanced Computer-based Manufacturing Systems for Defense Needs*, USC/Information Sciences Institute, RR-73-10, (in progress).

3 Balzer, Robert M., *Automatic Programming*, USC/Information Sciences Institute, RR-73-1, (draft).

4 ---, *CASAP: A Testbed for Program Flexibility*, USC/Information Sciences Institute, RR-73-5, (in progress).

5 ---, *Another Look at Program Organization*, USC/Information Sciences Institute, RR-73-6, (in progress).

6 ---, *Human Use of World Knowledge*, USC/Information Sciences Institute, RR-73-7, (in progress).

7 ---, *A Global View of Automatic Programming*, USC/Information Sciences Institute, RR-73-9, (in progress).

8 Ellis, Thomas O., Louis Gallenson, John F. Heafner, and John T. Melvin, *A Plan for Consolidation and Automation of Military Telecommunications on Oahu*, USC/Information Sciences Institute, RR-73-12, May 1973.

9 Igarashi, Shigeru, Ralph L. London, and David C. Luckham, "Automatic Program Verification I: Logical Basis and Its Implementation", *Artificial Intelligence Memo 200*, Stanford University, May 1973; also USC/Information Sciences Institute, RR-73-11, May 1973.

10 Kamrany, Nake M., *A Preliminary Analysis of the Economic Impact of Programmable Automation Upon Discrete Manufacturing Products*, USC/Information Sciences Institute, RR-73-4, (in progress).

11 Oestreicher, Donald R., *A Microprogramming Language for the MLP-900*, USC/Information Sciences Institute, RR-73-8, June 1973; will also appear in the Proceedings of the ACM Sigplan Sigmicro Interface Meeting, New York, May 30 - June 1, 1973.

12 Rosenberg, Jack, *A History of Numerical Control 1949 - 1972: The Technical Development, Transfer to Industry, and Assimilation*, USC/Information Sciences Institute, RR-73-3, (in progress).